Linux From Scratch

Version r11.1-135+

Publié le June 14th, 2022

Créé par Gerard Beekmans
Rédacteur en chef : Bruce Dubbs

Linux From Scratch: Version r11.1-135+: Publié le June 14th, 2022

par Créé par Gerard Beekmans et Rédacteur en chef : Bruce Dubbs

Copyright © 1999-2022 Gerard Beekmans

Copyright © 1999-2022, 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 viii
i. Avant-propos viii
ii. Public visé ix
iii. Architectures cibles de LFS ix
iv. Prérequis x
v. LFS et les standards x
vi. Raison de la présence des paquets dans le livre xii
vii. Typographie xvii
viii. Structure xviii
ix. Errata et annonces de sécurité xix
I. Introduction 1
1. Introduction 2
1.1. Comment construire un système LFS 2
1.2. Nouveautés depuis la dernière version 2
1.3. Historique des modifications 3
1.4. Ressources 6
1.5. Aide 6
II. Préparation à la construction 9
2. Préparation du système hôte 10
2.1. Introduction 10
2.2. Prérequis du système hôte 10
2.3. Les étapes de la construction de LFS 12
2.4. Création d'une nouvelle partition 13
2.5. Création d'un système de fichiers sur la partition 15
2.6. Définition de la variable $LFS 15
2.7. Montage de la nouvelle partition 16
3. Paquets et correctifs 18
3.1. Introduction 18
3.2. Tous les paquets 19
3.3. Correctifs requis 26
4. Dernières préparations 28
4.1. Introduction 28
4.2. Créer un ensemble limité de répertoire dans le système de fichiers LFS 28
4.3. Ajouter l'utilisateur LFS 28
4.4. Configurer l'environnement 29
4.5. À propos des SBU 31
4.6. À propos des suites de tests 32
III. Construction des outils croisés LFS et des outils temporaires 33
Informations préliminaires importantes xxxiv
i. Introduction xxxiv
ii. Notes techniques sur la chaîne d'outils xxxiv
iii. Instructions générales de compilation xxxviii
5. Compilation d'une chaîne d'outils croisée 40
5.1. Introduction 40
5.2. Binutils-2.38 — Passe 1 41
5.3. GCC-12.1.0 — Passe 1 43
5.4. Linux-5.18.3 API Headers 45
5.5. Glibc-2.35 46

5.6. Libstdc++ from GCC-12.1.0 49
6. Compilation croisée des outils temporaires 50
6.1. Introduction 50
6.2. M4-1.4.19 51
6.3. Ncurses-6.3 52
6.4. Bash-5.1.16 54
6.5. Coreutils-9.1 55
6.6. Diffutils-3.8 56
6.7. File-5.42 57
6.8. Findutils-4.9.0 58
6.9. Gawk-5.1.1 59
6.10. Grep-3.7 60
6.11. Gzip-1.12 61
6.12. Make-4.3 62
6.13. Patch-2.7.6 63
6.14. Sed-4.8 64
6.15. Tar-1.34 65
6.16. Xz-5.2.5 66
6.17. Binutils-2.38 — Passe 2 67
6.18. GCC-12.1.0 — Passe 2 68
7. Entrée dans le chroot et construction des outils temporaires supplémentaires 70
7.1. Introduction 70
7.2. Changement du propriétaire 70
7.3. Préparer les systèmes de fichiers virtuels du noyau 70
7.4. Entrer dans l'environnement chroot 71
7.5. Création des répertoires 72
7.6. Création des fichiers et des liens symboliques essentiels 72
7.7. Gettext-0.21 75
7.8. Bison-3.8.2 76
7.9. Perl-5.36.0 77
7.10. Python-3.10.5 78
7.11. Texinfo-6.8 79
7.12. Util-linux-2.38 80
7.13. Nettoyage et Sauvegarde du système temporaire 81
IV. Construction du système LFS 83
8. Installer les logiciels du système de base 84
8.1. Introduction 84
8.2. Gestion de paquets 84
8.3. Man-pages-5.13 89
8.4. Iana-Etc-20220524 90
8.5. Glibc-2.35 91
8.6. Zlib-1.2.12 98
8.7. Bzip2-1.0.8 99
8.8. Xz-5.2.5 101
8.9. Zstd-1.5.2 103
8.10. File-5.42 104
8.11. Readline-8.1.2 105
8.12. M4-1.4.19 106
8.13. Bc-5.3.1 107
8.14. Flex-2.6.4 108

8.15. Tcl-8.6.12 109
8.16. Expect-5.45.4 111
8.17. DejaGNU-1.6.3 112
8.18. Binutils-2.38 113
8.19. GMP-6.2.1 116
8.20. MPFR-4.1.0 118
8.21. MPC-1.2.1 119
8.22. Attr-2.5.1 120
8.23. Acl-2.3.1 121
8.24. Libcap-2.64 122
8.25. Shadow-4.11.1 123
8.26. GCC-12.1.0 127
8.27. Pkg-config-0.29.2 132
8.28. Ncurses-6.3 133
8.29. Sed-4.8 136
8.30. Psmisc-23.5 137
8.31. Gettext-0.21 138
8.32. Bison-3.8.2 140
8.33. Grep-3.7 141
8.34. Bash-5.1.16 142
8.35. Libtool-2.4.7 144
8.36. GDBM-1.23 145
8.37. Gperf-3.1 146
8.38. Expat-2.4.8 147
8.39. Inetutils-2.2 148
8.40. Less-590 150
8.41. Perl-5.36.0 151
8.42. XML::Parser-2.46 154
8.43. Intltool-0.51.0 155
8.44. Autoconf-2.71 156
8.45. Automake-1.16.5 158
8.46. OpenSSL-3.0.3 159
8.47. Kmod-29 161
8.48. Libelf de Elfutils-0.187 163
8.49. Libffi-3.4.2 164
8.50. Python-3.10.5 165
8.51. Wheel-0.37.1 167
8.52. Ninja-1.11.0 168
8.53. Meson-0.62.2 170
8.54. Coreutils-9.1 171
8.55. Check-0.15.2 176
8.56. Diffutils-3.8 177
8.57. Gawk-5.1.1 178
8.58. Findutils-4.9.0 179
8.59. Groff-1.22.4 180
8.60. GRUB-2.06 183
8.61. Gzip-1.12 185
8.62. IPRoute2-5.18.0 186
8.63. Kbd-2.5.0 188
8.64. Libpipeline-1.5.6 190

8.65. Make-4.3 191
8.66. Patch-2.7.6 192
8.67. Tar-1.34 193
8.68. Texinfo-6.8 194
8.69. Vim-8.2.5014 196
8.70. Eudev-3.2.11 199
8.71. Man-DB-2.10.2 201
8.72. Procps-4.0.0 204
8.73. Util-linux-2.38 206
8.74. E2fsprogs-1.46.5 211
8.75. Sysklogd-1.5.1 214
8.76. Sysvinit-3.04 215
8.77. À propos des symboles de débogage 216
8.78. Nettoyage 216
8.79. Nettoyage 218
9. Configuration du système 219
9.1. Introduction 219
9.2. LFS-Bootscripts-20220327 220
9.3. Manipulation des périphériques et modules 222
9.4. Gérer les périphériques 225
9.5. Configuration générale du réseau 228
9.6. Utiliser et configurer les scripts de démarrage de System V 230
9.7. Fichiers de démarrage du shell Bash 238
9.8. Créer le fichier /etc/inputrc 240
9.9. Créaction du fichier /etc/shells 241
10. Rendre le système LFS amorçable 243
10.1. Introduction 243
10.2. Créer le fichier /etc/fstab 243
10.3. Linux-5.18.3 245
10.4. Utiliser GRUB pour paramétrer le processus de démarrage 249
11. La fin 251
11.1. Fin 251
11.2. Enregistrez-vous 251
11.3. Redémarrer le système 251
11.4. Et maintenant ? 253
V. Annexes 254
A. Acronymes et termes 255
B. Remerciements 257
C. Dépendances 260
D. Scripts de démarrage et de sysconfig version-20220327 277
D.1. /etc/rc.d/init.d/rc 277
D.2. /lib/lsb/init-functions 280
D.3. /etc/rc.d/init.d/mountvirtfs 292
D.4. /etc/rc.d/init.d/modules 293
D.5. /etc/rc.d/init.d/udev 295
D.6. /etc/rc.d/init.d/swap 296
D.7. /etc/rc.d/init.d/setclock 297
D.8. /etc/rc.d/init.d/checkfs 298
D.9. /etc/rc.d/init.d/mountfs 300
D.10. /etc/rc.d/init.d/udev_retry 301

D.11. /etc/rc.d/init.d/cleanfs 303
D.12. /etc/rc.d/init.d/console 304
D.13. /etc/rc.d/init.d/localnet 306
D.14. /etc/rc.d/init.d/sysctl 307
D.15. /etc/rc.d/init.d/sysklogd 308
D.16. /etc/rc.d/init.d/network 309
D.17. /etc/rc.d/init.d/sendsignals 311
D.18. /etc/rc.d/init.d/reboot 312
D.19. /etc/rc.d/init.d/halt 312
D.20. /etc/rc.d/init.d/template 313
D.21. /etc/sysconfig/modules 314
D.22. /etc/sysconfig/createfiles 315
D.23. /etc/sysconfig/udev-retry 315
D.24. /sbin/ifup 315
D.25. /sbin/ifdown 318
D.26. /lib/services/ipv4-static 319
D.27. /lib/services/ipv4-static-route 320
E. Règles de configuration d'Udev 323
E.1. 55-lfs.rules 323
F. Licences LFS 324
F.1. Creative Commons License 324
F.2. The MIT License 328
Index 329

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 *https://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, Python
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 :	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.

- 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.

- Eudev

  Ce paquet est un gestionnaire de périphériques. Il contrôle de façon dynamique l'appartenance, les permissions, les noms et les liens symboliques de périphériques dans le répertoire /dev quand les périphériques sont ajoutés ou retirés du système.

- 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.

- 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. Il est requis pour construire Systemd, ainsi que de nombreux paquets de BLFS.

- 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. Ce paquet est requis par Meson.

- 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.

- Sysklogd

  Ce paquet contient des programmes de journalisation des messages système, tels que ceux donnés par le noyau ou les processus démons lorsque se produisent des événements inhabituels.

- Sysvinit

  Ce paquet fournit le programme init qui est le parent de tous les autres processus du système Linux.

- 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.

- Wheel

  This package contains a Python module that is the reference implementation of the Python wheel packaging standard.

- 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 taux 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.

https://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 et annonces de sécurité

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 *https://www.linuxfromscratch.org/lfs/errata/development/* 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.

En plus, les rédacteurs de Linux From Scratch maintiennent une liste des vulnérabilités de sécurité découvertes après la publication du livre. Pour vérifier s'il y a des vulnérabilités actives, visitez *https://www.linuxfromscratch.org/lfs/advisories/* avant de continuer votre construction. Vous devriez prendre en compte les annonces et prendre des mesures pour corriger les vulnérabilités au fur et à mesure de votre progression dans 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, ceci incluant 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 de l'installation d'une distribution séparée complète sur votre machine, vous pouvez 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 et tout ira à sa place, alors que vous vous embarquez pour l'aventure LFS.

1.2. Nouveautés depuis la dernière version

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

Mis à jour vers :

-
- Bc 5.3.1
- Coreutils-9.1
- E2fsprogs-1.46.5

- Expat-2.4.8
- File-5.42
- GCC-12.1.0
- Gzip-1.12
- IANA-Etc-20220524
- IPRoute2-5.18.0
- Kbd-2.5.0
- Libcap-2.64
- Libelf-0.187 (de elfutils)
- Libpipeline-1.5.6
- Libtool-2.4.7
- Linux-5.18.3
- Man-DB-2.10.2
- Meson-0.62.2
- Ninja-1.11.0
- Openssl-3.0.3
- Perl-5.36.0
- Procps-ng-4.0.0
- Psmisc-23.5
- Python-3.10.5
- SysVinit-3.04
- Tzdata-2022a
- Util-Linux-2.38
- Vim-8.2.5014
- Zlib-1.2.12

Ajoutés :

-
- Wheel-0.37.1
- xz-5.2.5-upstream_fix-1.patch
- zstd-1.5.2-upstream_fixes-1.patch

Supprimés :

-
- perl-5.34.0-upstream_fixes-1.patch
- systemd-250-kernel_5.17_fixes-1.patch
- systemd-250-upstream_fixes-1.patch

1.3. Historique des modifications

Il s'agit de la version r11.1-135+ du livre Linux From Scratch, datant du June 14th, 2022. 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, vérifiez la présence d'une nouvelle version sur l'un des miroirs via *https://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 :

- 2022-06-14
  - [bdubbs] - Update to python3-3.10.5. Fixes *#5070*.
  - [bdubbs] - Update to meson-0.62.2. Fixes *#5068*.
  - [bdubbs] - Update to kbd-2.5.0. Fixes *#5067*.
  - [bdubbs] - Update to file-5.42. Fixes *#5072*.
  - [bdubbs] - Update to linux-5.18.3. Fixes *#5069*.
  - [bdubbs] - Update to bc-5.3.1. Fixes *#5071*.
- 2022-05-29
  - [bdubbs] - Update to iana-etc-20220524. Addresses *#5006*.
  - [bdubbs] - Update to iproute2-5.18.0. Fixes *#5065*.
  - [bdubbs] - Update to linux-5.18.1. Fixes *#5061*.
  - [bdubbs] - Update to perl-5.36.0. Fixes *#5066*.
- 2022-05-29
  - [xry111] - Update to vim-8.2.5014 (security fixes). Addresses *#4500*.
  - [xry111] - Update to ninja-1.11.0. Fixes *#5062*.
  - [xry111] - Update to systemd-251. Fixes *#5064*.
- 2022-05-16
  - [bdubbs] - Update to linux-5.17.7. Fixes *#5059*.
  - [bdubbs] - Update to psmisc-23.5. Fixes *#5060*.
- 2022-05-01
  - [bdubbs] - Update to openssl-3.0.3. Fixes *#5057*.
- 2022-05-01
  - [bdubbs] - Change nobody/nogroup uid/gid to 65534.
  - [bdubbs] - Update to meson-0.62.1. Fixes *#5052*.
  - [bdubbs] - Update to libpipeline-1.5.6. Fixes *#5053*.
  - [bdubbs] - Update to elfutils-0.187. Fixes *#5054*.
  - [bdubbs] - Update to vim-8.2.4814. Addresses *#4500*.
  - [bdubbs] - Update to sysvinit-3.03. Fixes *#5046*.
  - [bdubbs] - Update to linux-5.17.5. Fixes *#5050*.
  - [bdubbs] - Update to gcc-11.3.0. Fixes *#5051*.
  - [bdubbs] - Update to coreutils-9.1. Fixes *#5048*.
  - [bdubbs] - Update to bc-5.2.4. Fixes *#5049*.
- 2022-04-15
  - [bdubbs] - Add wheel-0.37.1 (Python module).
- 2022-04-15
  - [ken] - Patch Xz for a security fix. Fixes *#5047*.

- [bdubbs] - Update to libcap-2.64. Fixes *#5044*.
- [bdubbs] - Update to linux-5.17.3. Fixes *#5043*.
- [bdubbs] - Update to gzip-1.12. Fixes *#5042*.

- 2022-03-31
  - [bdubbs] - Update to sysvinit-3.02. Fixes *#5041*.
  - [bdubbs] - Update to zlib-1.2.12 (Security Update). Fixes *#5040*.
  - [bdubbs] - Update to expat-2.4.8. Fixes *#5039*.
  - [bdubbs] - Update to Python-3.10.4. Fixes *#5037*.
  - [bdubbs] - Update to procps-ng-4.0.0. Fixes *#5036*.
  - [bdubbs] - Update to iproute2-5.17.0. Fixes *#5035*.
  - [bdubbs] - Update to meson-0.62.0. Fixes *#5034*.
  - [bdubbs] - Update to linux-5.17.1 (Security Update). Fixes *#5033*.
  - [bdubbs] - Update to util-linux-2.38. Fixes *#4997*.
- 2022-03-25
  - [pierre] - Update bootscripts to 20220324. Fixes *#5027*.
- 2022-03-20
  - [xry111] - Update to Python-3.10.3. Fixes *#5028*.
  - [xry111] - Update to libtool-2.4.7. Fixes *#5029*.
  - [xry111] - Update to linux-5.16.16. Fixes *#5030*.
  - [xry111] - Update to tzdata-2022a. Fixes *#5031*.
  - [xry111] - Update to man-db-2.10.2. Fixes *#5032*.
- 2022-03-16
  - [xry111] - Update to MarkupSafe-2.1.1. Fixes *#5025*.
- 2022-03-15
  - [bdubbs] - Update to openssl-3.0.2. Fixes *#5024*.
  - [bdubbs] - Update to meson-0.61.3. Fixes *#5023*.
  - [xry111] - Update to expat-2.4.7. Fixes *#5019*.
  - [xry111] - Update to bc-5.2.3. Fixes *#5020*.
  - [xry111] - Update to linux-5.16.14 (security fix). Fixes *#5021*.
  - [xry111] - Update to perl-5.34.1. Fixes *#5022*.
  - [xry111] - Update to vim-8.2.4567 (security fix). Addresses *#4500*.
- 2022-03-05
  - [xry111] - Stop changing the owner of `$LFS/source` to `lfs`. Fixes *#5018*.
  - [xry111] - Add zstd-1.5.2 upstream fixes patch.
- 2022-03-02
  - [xry111] - Update to meson-0.61.2. Fixes *#5013*.
  - [xry111] - Update to linux-5.16.12. Fixes *#5014*.
  - [xry111] - Update to MarkupSafe-2.1.0. Fixes *#5015*.

- [xry111] - Update to vim-8.2.4489 (security fixes). Addresses *#4500*.
- [xry111] - Build libstdc++ in GCC pass 2, and remove the seperate libstdc++ pass 2.
- [xry111] - Remove unneeded `--enable-64bit` for tcl.

- 01-03-2022
  - [bdubbs] — Publication de LFS-11.1.

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 diffusion sur *https://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 leur aide 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.libera.chat`. 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 *https://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*. Vous trouverez souvent la réponse à votre question ici. Dans le cas contraire, essayez de trouver la source du problème. Le guide suivant 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 diffusion sur *https://www.linuxfromscratch.org/search.html*.

Nous avons aussi une formidable communauté LFS, volontaire pour offrir une assistance via les listes de diffusion 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 diffusion.

Afin 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 que vous utilisez (dans ce cas, r11.1-135+)
- 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
- Indiquez 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 du 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 indique seulement que quelque chose s'est mal passé, pas *ce qui* s'est mal passé. C'est la section entière, comme dans l'exemple plus haut, 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, la présence des outils du système hôte nécessaires à la construction de LFS est vérifiée. Si besoin, ils sont installés avant de préparer la partition qui contiendra le système LFS. Nous créerons la partition elle-même, nous 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 des paquets distincts, ayant souvent la forme suivante :« <nom-du-paquet>-devel » ou « <nom-du-paquet>-dev ». Assurez-vous que ces derniers 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 matériel vers bash)
- **Binutils-2.13.1** (les versions ultérieures à 2.38 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)
- **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-4.8**, y compris le compilateur C++ **g++** (les versions ultérieures à 12.1.0 ne sont pas recommandées, car elles n'ont pas été testées). Les bibliothèques standards C et C++ (avec les en-têtes) doivent également être présentes pour que le compilateur C++ puisse construire les programmes.
- **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 aux Chapitre 5 et Chapitre 8, 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 récente. Il existe deux méthodes pour cela. Vous pouvez d'abord voir si votre distribution Linux fournit un paquet pour le noyau 3.2 ou pour un noyau plus récente. Si c'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) sont disponibles 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 dans ce livre. Les liens symboliques qui pointent vers d'autres logiciels (comme dash, mawk, etc) peuvent fonctionner, mais ils n'ont pas été testés, ou ne sont pas pris en charge par l'équipe de développement LFS. Iil est possible que ces liens demandent soit des déviations par rapport aux instructions, soit des correctifs supplémentaires de certains paquets.

Pour vérifier que votre système hôte possède toutes les versions nécessaires, exécutez la commande suivante :

```
cat > version-check.sh << "EOF"
#!/bin/bash
# Script simple pour afficher les numéros de version des outils de développement critiques
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 is `/usr/bin/yacc --version | head -n1`
else
  echo "yacc introuvable"
fi

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 is `/usr/bin/awk --version | head -n1`
else
  echo "awk introuvable"
fi
```

```
gcc --version | head -n1
g++ --version | head -n1
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 la construction de LFS à différentes étapes.

2.3.1. Chapitres 1 à 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.
- A few operations, from « Changing Ownership » to « Entering the Chroot Environment » must be done as the `root` user, with the LFS environment variable set for the `root` user.
- 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 7.3.1, « Monter et peupler /dev » et de la Section 7.3.2, « Monter les systèmes de fichiers virtuels du noyau ».

2.4. Création d'une nouvelle partition

Comme la plupart des autres systèmes d'exploitation, LFS est habituellement installé sur une partition dédiée. Pour la construction d'un système LFS, il est recommandé d'utiliser une partition vide disponible ou d'en créer une, si vous avez assez d'espace non partitionné.

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 vive (RAM) disponible pour les processus de compilation, il est conseillé d'utiliser une petite partition comme espace d'échange `swap`. Le noyau utilise cet espace pour stocker des données rarement utilisées et pour laisser plus de mémoire 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 possède 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 au manuel de `cfdisk(8)` ou de `fdisk(8)` si vous ne savez pas encore utiliser le programme.

Note

Pour les utilisateurs expérimentés, il est possible d'utiliser d'autres méthodes de partitionnement. Le nouveau système LFS peut s'installer sur du *RAID* logiciel ou sur un volume logique *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 novices de LFS.

Gardez en mémoire la désignation de la nouvelle partition (par exemple `sda5`). Ce livre y fera référence en tant que la partition LFS. Gardez également en tête la désignation de la partition `swap`. Ces noms vous seront utiles pour le fichier `/etc/fstab`.

2.4.1. Autres problèmes liés au 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 pour une partition d'échange est à peu près deux fois supérieure à la taille de la RAM physique, bien que ce soit rarement nécessaire. Si vous avez un espace 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 se trouver 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 assez importante et dépend de l'espace disque disponible.
- /usr – Dans LFS, `/bin`, `/lib` et `/sbin` sont des liens symboliques vers leur contrepartie dans `/usr`. Dans `/usr` contient tous les binaires requis pour lancer le système. Dans LFS, une partition séparée pour `/usr` n'est normalement pas requise. Si vous en avez quand même besoin, vous devrez la rendre assez grande pour contenir tous les programmes et bibliothèques du système. La partition racine peut être très petite (peut-être juste un gigaoctet) dans cette configuration, donc elle est parfaite pour un client léger ou une station de travail sans disque (où `/usr` est monté à partir d'un serveur distant). Cependant, vous devrez vous assurer qu'un initramfs (dont la construction n'est pas couverte par LFS) pourra démarrer le système avec une partition `/usr` séparée.

- /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, cinq à dix gigaoctets sont généralement adaptés.
- /tmp – Un répertoire /tmp séparé est rare, mais utile si vous configurez un client léger. Si vous l'utilisez, cette partition 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 comprise entre trente et cinquante gigaoctets permet d'avoir beaucoup de d'espace.

Vous devez spécifier toute partition que vous voulez voir montée automatiquement au démarrage dans `/etc/fstab`. La spécification des partitions sera expliquée en détails dans le Section 10.2, « Créer le fichier /etc/fstab ».

2.5. Création d'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 ext3 et ext4 sont les plus communs. 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
: est une mise à jour du type ext2 qui comprend un journal aidant à récupérer l'état de la partition en cas d'arrêt brutal. Ce type est communément employé dans une perspective généraliste.

ext4
: est la dernière version de types ext de systèmes de fichiers. Ce type 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 considère que le système de fichiers racine (/) est de type ext4. Pour créer un système de fichiers `ext4` sur la partition LFS, exécutez la commande suivante :

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

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

Si vous utilisez une partition de `swap` déjà existante, il n'est pas nécessaire de la formater. Afin d'utiliser une partition `swap` qui vient d'être créée, il sera nécessaire de l'initialiser à l'aide de la commande suivante :

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

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

2.6. Définition de la variable $LFS

Tout au long de ce livre, la variable d'environnement `LFS` est mentionnée à plusieurs reprises. Assurez-vous de toujours définir cette variable pendant le processus de construction de LFS. Cette variable 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 séparée, le répertoire défini sera le point de montage de la partition. Choisissez un répertoire et définissez la variable avec la commande suivante :

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

La définition de cette variable constitue un avantage dans des commandes telles que **mkdir -v $LFS/tools** que l'on peut taper littéralement. Le shell remplacera automatiquement « $LFS » par « /mnt/lfs » (ou par le répertoire défini dans la variable) quand il traitera la ligne de commande.

Attention

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

```
echo $LFS
```

Assurez-vous que la sortie affiche le chemin du répertoire dans lequel vous construisez votre système LFS, qui est `/mnt/lfs` si vous avez suivi l'exemple fourni. Si la sortie indique le mauvais répertoire, utilisez la commande précédemment indiquée dans cette page pour associer la variable `$LFS` au bon 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 le fichier `/root/.bash_profile` et d'y entrer la commande d'export mentionnée ci-dessus. De plus, pour tous les utilisateurs ayant besoin de la variable `LFS`, le shell indiqué dans le fichier `/etc/passwd` 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. Montage de 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 exécutant la commande suivante :

```
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 `/home`), montez-les en utilisant :

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

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`). Exécutez 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, soit modifier 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 exécutant 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 de base. Les numéros de versions affichés correspondent aux versions des logiciels qui sont connues pour fonctionner, et ce livre est basé sur leur utilisation. Nous vous recommandons fortement de ne pas utiliser de versions différentes, car les commandes de construction pour une version pourraient ne pas fonctionner avec une autre version, à moins que cette version différente ne soit mentionné dans un errata ou une information de sécurité. 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.

Pour certains paquets, l'archive de version et l'instantané du dépôt (Git ou SVN) pour cette version peuvent être publiés avec des noms similaires. Une archive de version contient des fichiers générés (par exemple, le script **configure** généré par **autoconf**) en plus du contenu de l'instantané du dépôt. Le livre utilise les archives de version chaque fois que cela est possible. Utiliser un instantané de dépôt au lieu d'une archive de version spécifiée par le livre causera des problèmes.

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 *https://www.linuxfromscratch.org/lfs/mirrors.html#files*.

Les paquets et les correctifs téléchargés doivent être stockés à un emplacement 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 *https://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
```

Note

The `wget-list` file mentioned above retrieves all packages for both the sysV and systemd versions of LFS. There are a total of five additional small packages not needed for the current book. The `md5sums` file mentioned below is specific to the current book.

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

Note

Read the *security advisories* before downloading packages to figure out if a newer version of any package should be used to avoid security vulnerabilities.

The upstreams may remove old releases, especially when these releases contain a security vulnerability. If one URL below is not reachable, you should read the security advisories first to figure out if a newer version (with the vulnerability fixed) should be used. If not, try to download the removed package from a mirror. Although it's possible to download an old release from a mirror even if this release has been removed because of a vulnerability, it's not recommended to use a release known to be vulnerable for building your system.

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

• Acl (2.3.1) — 348 Ko :
Page d'accueil : *https://savannah.nongnu.org/projects/acl*
Téléchargement : *https://download.savannah.gnu.org/releases/acl/acl-2.3.1.tar.xz*
Somme de contrôle MD5 : `95ce715fe09acca7c12d3306d0f076b2`

• Attr (2.5.1) — 456 Ko :
Page d'accueil : *https://savannah.nongnu.org/projects/attr*
Téléchargement : *https://download.savannah.gnu.org/releases/attr/attr-2.5.1.tar.gz*
Somme de contrôle MD5 : `ac1c5a7a084f0f83b8cace34211f64d8`

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

• Automake (1.16.5) — 1,565 Ko:
Page d'accueil : *https://www.gnu.org/software/automake/*
Téléchargement : *https://ftp.gnu.org/gnu/automake/automake-1.16.5.tar.xz*
Somme de contrôle MD5 : `4017e96f89fca45ca946f1c5db6be714`
Somme de contrôle SHA256 : `80facc09885a57e6d49d06972c0ae1089c5fa8f4d4c7cfe5baea58e5085f136d`

• Bash (5.1.16) — 10,277 Ko:
Page d'accueil : *https://www.gnu.org/software/bash/*
Téléchargement : *https://ftp.gnu.org/gnu/bash/bash-5.1.16.tar.gz*
Somme de contrôle MD5 : `c17b20a09fc38d67fb303aeb6c130b4e`

• Bc (5.3.1) — 434 Ko:
Page d'accueil : *https://git.yzena.com/gavin/bc*
Téléchargement : *https://github.com/gavinhoward/bc/releases/download/5.3.1/bc-5.3.1.tar.xz*
Somme de contrôle MD5 : `10dbd6425f22987ad7ea6d68d55f3680`

• Binutils (2.38) — 23,098 Ko:
Page d'accueil : *https://www.gnu.org/software/binutils/*
Téléchargement : *https://ftp.gnu.org/gnu/binutils/binutils-2.38.tar.xz*
Somme de contrôle MD5 : `6e39cad1bb414add02b5b1169c18fdc5`

• Bison (3.8.2) — 2,752 Ko:
Page d'accueil : *https://www.gnu.org/software/bison/*
Téléchargement : *https://ftp.gnu.org/gnu/bison/bison-3.8.2.tar.xz*
Somme de contrôle MD5 : `c28f119f405a2304ff0a7ccdcc629713`

• 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 (9.1) — 5,570 Ko:
Page d'accueil : *https://www.gnu.org/software/coreutils/*
Téléchargement : *https://ftp.gnu.org/gnu/coreutils/coreutils-9.1.tar.xz*
Somme de contrôle MD5 : `8b1ca4e018a7dce9bb937faec6618671`

• DejaGNU (1.6.3) — 608 Ko:
Page d'accueil : *https://www.gnu.org/software/dejagnu/*
Téléchargement : *https://ftp.gnu.org/gnu/dejagnu/dejagnu-1.6.3.tar.gz*
Somme de contrôle MD5 : `68c5208c58236eba447d7d6d1326b821`

• Diffutils (3.8) — 1,548 Ko:
Page d'accueil : *https://www.gnu.org/software/diffutils/*
Téléchargement : *https://ftp.gnu.org/gnu/diffutils/diffutils-3.8.tar.xz*
Somme de contrôle MD5 : `6a6b0fdc72acfe3f2829aab477876fbc`

• E2fsprogs (1.46.5) — 9,307 Ko:
Page d'accueil : *http://e2fsprogs.sourceforge.net/*
Téléchargement : *https://downloads.sourceforge.net/project/e2fsprogs/e2fsprogs/v1.46.5/e2fsprogs-1.46.5.tar.gz*
Somme de contrôle MD5 : `3da91854c960ad8a819b48b2a404eb43`

• Elfutils (0.187) — 9,024 Ko:
Page d'accueil : *https://sourceware.org/elfutils/*
Téléchargement : *https://sourceware.org/ftp/elfutils/0.187/elfutils-0.187.tar.bz2*
Somme de contrôle MD5 : `cc04f07b53a71616b22553c0a458cf4b`

• Eudev (3.2.11) — 2,075 Ko:
Téléchargement : *https://github.com/eudev-project/eudev/releases/download/v3.2.11/eudev-3.2.11.tar.gz*
Somme de contrôle MD5 : `417ba948335736d4d81874fba47a30f7`

• Expat (2.4.8) — 444 Ko :
Page d'accueil : *https://libexpat.github.io/*
Téléchargement : *https://prdownloads.sourceforge.net/expat/expat-2.4.8.tar.xz*
Somme de contrôle MD5 : `0584a7318a4c007f7ec94778799d72fe`

• 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 (5.42) — 1,080 Ko:
Page d'accueil : *https://www.darwinsys.com/file/*
Téléchargement : *https://astron.com/pub/file/file-5.42.tar.gz*
Somme de contrôle MD5 : `4d4f70c3b08a8a70d8baf67f085d7e92`

• Findutils (4.9.0) — 1,999 Ko:
Page d'accueil : *https://www.gnu.org/software/findutils/*
Téléchargement : *https://ftp.gnu.org/gnu/findutils/findutils-4.9.0.tar.xz*
Somme de contrôle MD5 : `4a4a547e888a944b2f3af31d789a1137`

• 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 (5.1.1) — 3,075 Ko:
Page d'accueil : *https://www.gnu.org/software/gawk/*
Téléchargement : *https://ftp.gnu.org/gnu/gawk/gawk-5.1.1.tar.xz*
Somme de contrôle MD5 : `83650aa943ff2fd519b2abedf8506ace`

• GCC (12.1.0) — 78,875 Ko:
Page d'accueil : *https://gcc.gnu.org/*
Téléchargement : *https://ftp.gnu.org/gnu/gcc/gcc-12.1.0/gcc-12.1.0.tar.xz*
Somme de contrôle MD5 : `ed45b55ee859ada4b25a1e76e0c4d966`
Somme de contrôle SHA256 : `62fd634889f31c02b64af2c468f064b47ad1ca78411c45abe6ac4b5f8dd19c7b`

• GDBM (1.23) — 1,092 Ko :
Page d'accueil : *https://www.gnu.org/software/gdbm/*
Téléchargement : *https://ftp.gnu.org/gnu/gdbm/gdbm-1.23.tar.gz*
Somme de contrôle MD5 : `8551961e36bf8c70b7500d255d3658ec`

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

• Glibc (2.35) — 17,741 Ko :
Page d'accueil : *https://www.gnu.org/software/libc/*
Téléchargement : *https://ftp.gnu.org/gnu/glibc/glibc-2.35.tar.xz*
Somme de contrôle MD5 : `dd571c67d85d89d7f60b854a4e207423`

• GMP (6.2.1) — 1,980 Ko:
Page d'accueil : *https://www.gnu.org/software/gmp/*
Téléchargement : *https://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 : *https://www.gnu.org/software/gperf/*
Téléchargement : *https://ftp.gnu.org/gnu/gperf/gperf-3.1.tar.gz*
Somme de contrôle MD5 : `9e251c0a618ad0824b51117d5d9db87e`

• Grep (3.7) — 1,603 Ko:
Page d'accueil : *https://www.gnu.org/software/grep/*
Téléchargement : *https://ftp.gnu.org/gnu/grep/grep-3.7.tar.xz*
Somme de contrôle MD5 : `7c9cca97fa18670a21e72638c3e1dabf`

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

• GRUB (2.06) — 6,428 Ko:
Page d'accueil : *https://www.gnu.org/software/grub/*
Téléchargement : *https://ftp.gnu.org/gnu/grub/grub-2.06.tar.xz*
Somme de contrôle MD5 : `cf0fd928b1e5479c8108ee52cb114363`

• Gzip (1.12) — 807 Ko:
Page d'accueil : *https://www.gnu.org/software/gzip/*
Téléchargement : *https://ftp.gnu.org/gnu/gzip/gzip-1.12.tar.xz*
Somme de contrôle MD5 : `9608e4ac5f061b2a6479dc44e917a5db`

• Iana-Etc (20220524) — 582 Ko:
Page d'accueil : *https://www.iana.org/protocols*
Téléchargement : *https://github.com/Mic92/iana-etc/releases/download/20220524/iana-etc-20220524.tar.gz*
Somme de contrôle MD5 : `a79e10f6c520a237a28522a25a7a4496`

• Inetutils (2.2) — 1,494 Ko:
Page d'accueil : *https://www.gnu.org/software/inetutils/*
Téléchargement : *https://ftp.gnu.org/gnu/inetutils/inetutils-2.2.tar.xz*
Somme de contrôle MD5 : `de8c1b49cbde2b30e481c61c65357ad4`
Somme de contrôle SHA256 : `01b9a4bc73a47e63f6e8a07b76122d9ad2a2e46ebf14870e9c91d660b5647a22`

• 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 (5.18.0) — 860 Ko:
Page d'accueil : *https://www.kernel.org/pub/linux/utils/net/iproute2/*
Téléchargement : *https://www.kernel.org/pub/linux/utils/net/iproute2/iproute2-5.18.0.tar.xz*
Somme de contrôle MD5 : `f666bab8ccf9c972938a0c0d8a4e6a68`

• Kbd (2.5.0) — 1,456 Ko:
Page d'accueil : *https://kbd-project.org/*
Téléchargement : *https://www.kernel.org/pub/linux/utils/kbd/kbd-2.5.0.tar.xz*
Somme de contrôle MD5 : `9b739cfd88e854f23d906d1990341fda`

• Kmod (29) — 548 Ko:
Téléchargement : *https://www.kernel.org/pub/linux/utils/kernel/kmod/kmod-29.tar.xz*
Somme de contrôle MD5 : `e81e63acd80697d001c8d85c1acb38a0`

• Less (590) — 348 Ko:
Page d'accueil : *https://www.greenwoodsoftware.com/less/*
Téléchargement : *https://www.greenwoodsoftware.com/less/less-590.tar.gz*
Somme de contrôle MD5 : `f029087448357812fba450091a1172ab`

• LFS-Bootscripts (20220327) — 33 Ko:
Téléchargement : *https://www.linuxfromscratch.org/lfs/downloads/development/lfs-bootscripts-20220327.tar.xz*
Somme de contrôle MD5 : `5807a8b3c9f8499379797e718485b29a`

• Libcap (2.64) — 172 Ko :
Page d'accueil : *https://sites.google.com/site/fullycapable/*
Téléchargement : *https://www.kernel.org/pub/linux/libs/security/linux-privs/libcap2/libcap-2.64.tar.xz*
Somme de contrôle MD5 : `14fb1fa596f58596bdef0bb5033d2586`

• Libffi (3.4.2) — 1,320 Ko:
Page d'accueil : *https://sourceware.org/libffi/*
Téléchargement : *https://github.com/libffi/libffi/releases/download/v3.4.2/libffi-3.4.2.tar.gz*
Somme de contrôle MD5 : `294b921e6cf9ab0fbaea4b639f8fdbe8`

• Libpipeline (1.5.6) — 954 Ko :
Page d'accueil : *http://libpipeline.nongnu.org/*
Téléchargement : *https://download.savannah.gnu.org/releases/libpipeline/libpipeline-1.5.6.tar.gz*
Somme de contrôle MD5 : `829c9ba46382b0b3e12dd11fcbc1bb27`

• Libtool (2.4.7) — 996 Ko:
Page d'accueil : *https://www.gnu.org/software/libtool/*
Téléchargement : *https://ftp.gnu.org/gnu/libtool/libtool-2.4.7.tar.xz*
Somme de contrôle MD5 : `2fc0b6ddcd66a89ed6e45db28fa44232`

• Linux (5.18.3) — 126,817 Ko:
Page d'accueil : *https://www.kernel.org/*
Téléchargement : *https://www.kernel.org/pub/linux/kernel/v5.x/linux-5.18.3.tar.xz*
Somme de contrôle MD5 : `a8db20740f7398af70543a645a4cd7ae`

Note

Le noyau Linux est régulièrement mis à jour, souvent suite à la découverte de nouvelles failles de sécurité. Vous devriez utiliser la version 5.18.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 (1.4.19) — 1,617 Ko:
Page d'accueil : *https://www.gnu.org/software/m4/*
Téléchargement : *https://ftp.gnu.org/gnu/m4/m4-1.4.19.tar.xz*
Somme de contrôle MD5 : `0d90823e1426f1da2fd872df0311298d`

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

• Man-DB (2.10.2) — 1,860 Ko:
Page d'accueil : *https://www.nongnu.org/man-db/*
Téléchargement : *https://download.savannah.gnu.org/releases/man-db/man-db-2.10.2.tar.xz*
Somme de contrôle MD5 : `e327f7af3786d15e5851658ae7ef47ed`

• Man-pages (5.13) — 1,752 Ko:
Page d'accueil : *https://www.kernel.org/doc/man-pages/*
Téléchargement : *https://www.kernel.org/pub/linux/docs/man-pages/man-pages-5.13.tar.xz*
Somme de contrôle MD5 : `3ac24e8c6fae26b801cb87ceb63c0a30`

• Meson (0.62.2) — 1,991 Ko:
Page d'accueil : *https://mesonbuild.com*
Téléchargement : *https://github.com/mesonbuild/meson/releases/download/0.62.2/meson-0.62.2.tar.gz*
Somme de contrôle MD5 : `e6521b43730901bdd67afa2adefa64a3`

• 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 : *https://ftp.gnu.org/gnu/mpfr/mpfr-4.1.0.tar.xz*
Somme de contrôle MD5 : `bdd3d5efba9c17da8d83a35ec552baef`

• Ncurses (6.3) — 3,500 Ko:
Page d'accueil : *https://www.gnu.org/software/ncurses/*
Téléchargement : *https://invisible-mirror.net/archives/ncurses/ncurses-6.3.tar.gz*
Somme de contrôle MD5 : `a2736befde5fee7d2b7eb45eb281cdbe`

• Ninja (1.11.0) — 228 Ko:
Page d'accueil : *https://ninja-build.org/*
Téléchargement : *https://github.com/ninja-build/ninja/archive/v1.11.0/ninja-1.11.0.tar.gz*
Somme de contrôle MD5 : `7d1a1a2f5cdc06795b3054df5c17d5ef`

• OpenSSL (3.0.3) — 14,706 Ko:
Page d'accueil : *https://www.openssl.org/*
Téléchargement : *https://www.openssl.org/source/openssl-3.0.3.tar.gz*
Somme de contrôle MD5 : `07e9048cd2b1a22668b13ec9cd093cb5`

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

• Perl (5.36.0) — 12,746 Ko:
Page d'accueil : *https://www.perl.org/*
Téléchargement : *https://www.cpan.org/src/5.0/perl-5.36.0.tar.xz*
Somme de contrôle MD5 : `826e42da130011699172fd655e49cfa2`

• 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 (4.0.0) — 979 Ko:
Page d'accueil : *https://sourceforge.net/projects/procps-ng*
Téléchargement : *https://sourceforge.net/projects/procps-ng/files/Production/procps-ng-4.0.0.tar.xz*
Somme de contrôle MD5 : `eedf93f2f6083afb7abf72188018e1e5`

• Psmisc (23.5) — 395 Ko:
Page d'accueil : *https://gitlab.com/psmisc/psmisc*
Téléchargement : *https://sourceforge.net/projects/psmisc/files/psmisc/psmisc-23.5.tar.xz*
Somme de contrôle MD5 : `014f0b5d5ab32478a2c57812ad01e1fb`

• Python (3.10.5) — 18,908 Ko:
Page d'accueil : *https://www.python.org/*
Téléchargement : *https://www.python.org/ftp/python/3.10.5/Python-3.10.5.tar.xz*
Somme de contrôle MD5 : `f05727cb3489aa93cd57eb561c16747b`

• Documentation Python (3.10.5) — 7,112 Ko:
Téléchargement : *https://www.python.org/ftp/python/doc/3.10.5/python-3.10.5-docs-html.tar.bz2*
Somme de contrôle MD5 : `46a8c3285c6ca87736bc9aad2d8519ba`

• Readline (8.1.2) — 2,923 Ko:
Page d'accueil : *https://tiswww.case.edu/php/chet/readline/rltop.html*
Téléchargement : *https://ftp.gnu.org/gnu/readline/readline-8.1.2.tar.gz*
Somme de contrôle MD5 : `12819fa739a78a6172400f399ab34f81`

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

• Shadow (4.11.1) — 1,618 Ko:
Page d'accueil : *https://shadow-maint.github.io/shadow/*
Téléchargement : *https://github.com/shadow-maint/shadow/releases/download/v4.11.1/shadow-4.11.1.tar.xz*
Somme de contrôle MD5 : `5a95ec069aa91508167d02fecafaa912`

• Sysklogd (1.5.1) — 88 Ko:
Page d'accueil : *https://www.infodrom.org/projects/sysklogd/*
Téléchargement : *https://www.infodrom.org/projects/sysklogd/download/sysklogd-1.5.1.tar.gz*
Somme de contrôle MD5 : `c70599ab0d037fde724f7210c2c8d7f8`

• Sysvinit (3.04) — 216 Ko:
Page d'accueil : *https://savannah.nongnu.org/projects/sysvinit*
Téléchargement : *https://download.savannah.gnu.org/releases/sysvinit/sysvinit-3.04.tar.xz*
Somme de contrôle MD5 : `9a00e5f15dd2f038f10feee50677ebff`

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

• Tcl (8.6.12) — 10,112 Ko:
Page d'accueil : *http://tcl.sourceforge.net/*
Téléchargement : *https://downloads.sourceforge.net/tcl/tcl8.6.12-src.tar.gz*
Somme de contrôle MD5 : `87ea890821d2221f2ab5157bc5eb885f`

• Documentation de Tcl (8.6.12) — 1,176 Ko :
Téléchargement : *https://downloads.sourceforge.net/tcl/tcl8.6.12-html.tar.gz*
Somme de contrôle MD5 : `a0d1a5b60bbb68f2f0bd3066a19c527a`

• Texinfo (6.8) — 4,848 Ko:
Page d'accueil : *https://www.gnu.org/software/texinfo/*
Téléchargement : *https://ftp.gnu.org/gnu/texinfo/texinfo-6.8.tar.xz*
Somme de contrôle MD5 : `a91b404e30561a5df803e6eb3a53be71`

• Time Zone Data (2022a) — 416 Ko :
Page d'accueil : *https://www.iana.org/time-zones*
Téléchargement : *https://www.iana.org/time-zones/repository/releases/tzdata2022a.tar.gz*
Somme de contrôle MD5 : `081662184f3b902f8e5d96816568432e`

• Udev-lfs Archive Tar (udev-lfs-20171102) — 11 Ko :
Téléchargement : *https://anduin.linuxfromscratch.org/LFS/udev-lfs-20171102.tar.xz*
Somme de contrôle MD5 : `27cd82f9a61422e186b9d6759ddf1634`

• Util-linux (2.38) — 7,177 Ko:
Page d'accueil : *https://git.kernel.org/pub/scm/utils/util-linux/util-linux.git/*
Téléchargement : *https://www.kernel.org/pub/linux/utils/util-linux/v2.38/util-linux-2.38.tar.xz*
Somme de contrôle MD5 : `fffdc126809a6d4c1789a9efc1b96623`

• Vim (8.2.5014) — 10,568 Ko:
Page d'accueil : *https://www.vim.org*
Téléchargement : *https://anduin.linuxfromscratch.org/LFS/vim-8.2.5014.tar.xz*
Somme de contrôle MD5 : `5b27e59fa53b74611422af6fa4149957`

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

• Wheel (0.37.1) - 65 Ko:
Home page: *https://pypi.org/project/wheel/*
Download: *https://anduin.linuxfromscratch.org/LFS/wheel-0.37.1.tar.gz*
MD5 sum: `f490f1399e5903706cb1d4fbed9ecb28`

• 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 (1.2.12) — 1259 Ko:
Page d'accueil : *https://www.zlib.net/*
Téléchargement : *https://zlib.net/zlib-1.2.12.tar.xz*
Somme de contrôle MD5 : `28687d676c04e7103bb6ff2b9694c471`

• Zstd (1.5.2) — 1,892 Ko :
Page d'accueil : *https://facebook.github.io/zstd/*
Téléchargement : *https://github.com/facebook/zstd/releases/download/v1.5.2/zstd-1.5.2.tar.gz*
Somme de contrôle MD5 : `072b10f71f5820c24761a65f31f43e73`

Taille totale de ces paquets : environ 447 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 :

• Correctif LTO de Binutils - 3.5 Ko :
Téléchargement : *https://www.linuxfromscratch.org/patches/lfs/development/binutils-2.38-lto_fix-1.patch*
Somme de contrôle MD5 : `3df11b6123d5bbdb0fc83862a003827a`

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

• Coreutils Correctif pour l'internationalisation — 166 Ko:
Téléchargement : *https://www.linuxfromscratch.org/patches/lfs/development/coreutils-9.1-i18n-1.patch*
Somme de contrôle MD5 : `c1ac7edf095027460716577633da9fc5`

• Glibc correctif FHS — 2.8 Ko :
Téléchargement : *https://www.linuxfromscratch.org/patches/lfs/development/glibc-2.35-fhs-1.patch*
Somme de contrôle MD5 : `9a5997c3452909b1769918c759eff8a2`

• Correctif réparant Kdb Backspace/Delete — 12 Ko :
Téléchargement : *https://www.linuxfromscratch.org/patches/lfs/development/kbd-2.5.0-backspace-1.patch*
Somme de contrôle MD5 : `f75cca16a38da6caa7d52151f7136895`

• Sysvinit Correctif consolidé — 2.4 Ko:
Téléchargement : *https://www.linuxfromscratch.org/patches/lfs/development/sysvinit-3.04-consolidated-1.patch*
Somme de contrôle MD5 : `4900322141d493e74020c9cf437b2cdc`

• Xz Upstream Fix Patch - 3.7 Ko:
Download: *https://www.linuxfromscratch.org/patches/lfs/development/xz-5.2.5-upstream_fix-1.patch*
MD5 sum: `584c72ea3d8f7502b2770670f3090788`

• Zstd Upstream Fixes Patch - 4 Ko:
Download: *https://www.linuxfromscratch.org/patches/lfs/development/zstd-1.5.2-upstream_fixes-1.patch*
MD5 sum: `a7e576e3f87415fdf388392b257cdcf3`

Taille totale de ces correctifs : environ 196 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 *https://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 (ainsi que glibc et libstdc++ dans le Chapitre 5) 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/{etc,var} $LFS/usr/{bin,lib,sbin}

for i in bin lib sbin; do
  ln -sv usr/$i $LFS/$i
done

case $(uname -m) in
  x86_64) mkdir -pv $LFS/lib64 ;;
esac
```

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

Note

In some host systems, the following command does not complete properly and suspends the login to the `lfs` user to the background. If the prompt "lfs:~$" does not appear immediately, entering the **fg** command will fix the issue.

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** qu'il doit 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) puis `.bash_profile`. La commande **exec env -i.../bin/bash** dans le fichier `.bash_profile` remplace le shell en cours par un nouveau shell ayant un environnement complètement vide en dehors des 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 permet d'obtenir 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 le `PATH` lorsqu'un programme doit être exécuté. Ainsi, le shell trouvera les nouveaux outils compilés dans `$LFS/tools/bin` dès qu'ils sont disponibles et sans se rappeler la version précédente du même programme fourni par la distribution hôte, dans `/usr/bin` ou `/bin`.

`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 finiront avec les droits 644 et les répertoires avec les droits 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-5.1.16 » 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 à l'avance 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 d'estimer la durée précise. 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, y compris de 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. Dans certains cas, l'étape make échouera. 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 » indiquant que tout a été compilé correctement. Une suite de tests réussissant l'ensemble des vérifications prouve généralement que le paquet fonctionne comme le développeur le souhaitait. 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 les développeurs sont conscients des raisons de ces échecs et les ont estimés non critique. Consultez les traces sur *https://www.linuxfromscratch.org/lfs/build-logs/development/* 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

Note

Le livre LFS n'est pas, et ne contient pas, un tutoriel générique sur la construction d'une chaine d'outils croisée (ou native). N'utilisez pas les commandes de ce livre pour une chaine d'outils croisée autrement que pour construire LFS, à moins que vous ne compreniez vraiment ce que vous faites.

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

Étape	Construction	Hôte	Cible	Action
				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 un nom à quatre composants s'appelle un « triplet ». 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*.

Faites également 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** q 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, dans LFS, 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

Étape	Construction	Hôte	Cible	Action
3	lfs	lfs	lfs	reconstruire et tester cc-lfs avec lui-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.

      Aucune méthode qui permette de placer l'arborescence du code source en place, en dehors de l'extraction d'une archive, n'est prise en charge. En particulier, l'utilisation de **cp -R** pour copier l'arborescence ailleurs peut détruire les liens et l'horodatage des fichiers et résulter en un échec à la construction.
   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-2.38 — 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: 620 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. En effet, 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 configuration :

`--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 systèmes cibles nécessaires.

`--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-12.1.0 — Passe 1

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

Temps de construction approximatif: 11 SBU
Espace disque requis: 3.8 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=2.35 \
    --with-sysroot=$LFS        \
    --with-newlib              \
    --without-headers          \
    --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 configuration :

`--with-glibc-version=2.35`
Cette option spécifie la version de glibc qui sera utilisée sur la cible. Ce n'est pas lié à la libc de la distribution hôte parce que tout ce qui est compilé par gcc passe 1 sera lancé dans l'environnement chroot, qui est isolé de la glibc de la distribution 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.

`--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.h
```

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

5.4. Linux-5.18.3 API Headers

Les en-têtes de l'API du noyau Linux (dans linux-5.18.3.tar.xz) expose les API du noyau à Glibc.

Temps de construction approximatif: 0.1 SBU
Espace disque requis: 1.2 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-2.35

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: 4.3 SBU
Espace disque requis: 818 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
```

Note

La commande ci-dessus est correcte. La commande **ln** a plusieurs versions syntaxiques, donc assurez-vous de vérifier **info coreutils ln** et `ln(1)` avant de rapporter ce que vous pensez être une erreur.

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-2.35-fhs-1.patch
```

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

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

Ensure that the **ldconfig** and **sln** utilities are installed into `/usr/sbin`:

```
echo "rootsbindir=/usr/sbin" > configparms
```

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=/usr/lib
```

Voici la signification des options de configuration :

`--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 en prenant en charge les noyaux Linux 3.2 et ultérieurs. 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=/usr/lib`

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

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 ».

Corrigez un chemin codé en dur vers le chargeur d'exécutable dans le script **ldd** :

```
sed '/RTLDLIST=/s@/usr@@g' -i $LFS/usr/bin/ldd
```

Attention

À ce stade, 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 fonctionne pas bien. Examinez et reconstituez vos étapes pour trouver d'où le problème vient 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/12.1.0/install-tools/mkheaders
```

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

5.6. Libstdc++ from GCC-12.1.0

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: 818 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-12.1.0`.

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++/12.1.0
```

Voici la signification des options de configuration :

`--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/$LFS_TGT/include/c++/12.1.0`
Cela spécifie le répertoire d'installation des fichiers d'en-tête. Comme libstdc++ est la bibliothèque standard du C++ dans LFS, ce répertoire doit correspondre à l'emplacement où le compilateur C++ (**$LFS_TGT-g++**) cherche les fichiers d'en-tête C++ standards. Dans une construction standard, cette information est automatiquement passée aux options **configure** de libstdc++ à partir du répertoire de plus haut niveau. Dans notre cas, cette information doit être passée explicitement. Le compilateur C++ ajoutera le chemin sysroot `$LFS` (spécifié à la construction de GCC passe 1) au début du chemin de recherche des en-têtes, pour qu'il recherche effectivement dans `$LFS/tools/$LFS_TGT/include/c++/12.1.0`. La combinaison de la variable *`DESTDIR`* (dans la commande **make install** ci-dessous) et ce paramètre s'assure d'installer les en-têtes à cet emplacement.

Compilez libstdc++ en lançant :

```
make
```

Installez la bibliothèque :

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

Les détails sur ce paquet sont disponibles 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. M4-1.4.19

Le paquet M4 contient un processeur de macros.

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

6.2.1. Installation de M4

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 sur ce paquet sont disponibles dans Section 8.12.2, « Contenu de M4. »

6.3. Ncurses-6.3

Le paquet Ncurses contient les bibliothèques pour gérer les écrans type caractère indépendamment des terminaux.

Temps de construction approximatif: 0.7 SBU
Espace disque requis: 50 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             \
            --disable-stripping          \
            --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**.

`--disable-stripping`
Ce paramètre évite que le système de construction n'efface les programmes avec le **strip** de l'hôte. L'utilisation des outils hôtes sur les programmes compilés de manière croisée peut causer des échecs.

`--enable-widec`
Cette option amène les bibliothèques « wide-character » (comme `libncursesw.so.6.3`) à être compilée au lieu de celles normales (comme `libncurses.so.6.3`). 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
```

Remove an unneeded static library not handled by **configure**:

```
rm -v $LFS/usr/lib/libncurses++w.a
```

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.

Les détails sur ce paquet sont disponibles dans Section 8.28.2, « Contenu de Ncurses. »

6.4. Bash-5.1.16

Le paquet Bash contient le shell Bourne-Again.

Temps de construction approximatif: 0.4 SBU
Espace disque requis: 64 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 configuration :

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

Compilez le paquet :

```
make
```

Installez le paquet :

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

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

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

Les détails sur ce paquet sont disponibles dans Section 8.34.2, « Contenu de Bash. »

6.5. Coreutils-9.1

Le paquet Coreutils contient des outils d'affichage de de configuration des caractéristiques de base d'un système.

Temps de construction approximatif: 0.6 SBU
Espace disque requis: 158 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`
Ceci fait que le binaire **hostname** sera compilé et installé – ceci est désactivé par défaut mais est requis par la suite de tests de Perl.

Compilez le paquet :

```
make
```

Installez le paquet :

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

Move programs to their final expected locations. Although this is not necessary in this temporary environment, we must do so because some programs hardcode executable locations:

```
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/man8/chroot.8
sed -i 's/"1"/"8"/'                    $LFS/usr/share/man/man8/chroot.8
```

Les détails sur ce paquet sont disponibles dans Section 8.54.2, « Contenu de Coreutils. »

6.6. Diffutils-3.8

Le paquet Diffutils contient des programmes qui affichent les différences entre fichiers ou répertoires.

Temps de construction approximatif: 0.2 SBU
Espace disque requis: 27 Mo

6.6.1. Installation de Diffutils

Préparez la compilation de Diffutils :

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

Compilez le paquet :

```
make
```

Installez le paquet :

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

Les détails sur ce paquet sont disponibles dans Section 8.56.2, « Contenu de Diffutils. »

6.7. File-5.42

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

Temps de construction approximatif: 0.1 SBU
Espace disque requis: 32 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 sur ce paquet sont disponibles dans Section 8.10.2, « Contenu de File. »

6.8. Findutils-4.9.0

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                   \
            --localstatedir=/var/lib/locate \
            --host=$LFS_TGT                 \
            --build=$(build-aux/config.guess)
```

Compilez le paquet :

```
make
```

Installez le paquet :

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

Les détails sur ce paquet sont disponibles dans Section 8.58.2, « Contenu de Findutils. »

6.9. Gawk-5.1.1

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

Temps de construction approximatif: 0.2 SBU
Espace disque requis: 45 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=$LFS_TGT \
            --build=$(build-aux/config.guess)
```

Compilez le paquet :

```
make
```

Installez le paquet :

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

Les détails sur ce paquet sont disponibles dans Section 8.57.2, « Contenu de Gawk. »

6.10. Grep-3.7

Le paquet Grep contient des programmes de recherche du 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
```

Compilez le paquet :

```
make
```

Installez le paquet :

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

Les détails sur ce paquet sont disponibles dans Section 8.33.2, « Contenu de Grep. »

6.11. Gzip-1.12

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

Temps de construction approximatif: 0.1 SBU
Espace disque requis: 11 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
```

Les détails sur ce paquet sont disponibles dans Section 8.61.2, « 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: 15 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 sur ce paquet sont disponibles dans Section 8.65.2, « 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 ») généralement créé par le programme **diff**.

Temps de construction approximatif: 0.1 SBU
Espace disque requis: 12 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 sur ce paquet sont disponibles dans Section 8.66.2, « 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: 20 Mo

6.14.1. Installation de Sed

Préparez la compilation de Sed :

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

Compilez le paquet :

```
make
```

Installez le paquet :

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

Les détails sur ce paquet sont disponibles dans 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: 38 Mo

6.15.1. Installation de Tar

Préparez la compilation de Tar :

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

Compilez le paquet :

```
make
```

Installez le paquet :

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

Les détails sur ce paquet sont disponibles dans 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: 15 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
```

Les détails sur ce paquet sont disponibles dans Section 8.8.2, « Contenu de Xz. »

6.17. Binutils-2.38 — 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: 520 Mo

6.17.1. Installation de Binutils

Binutils contient une copie obsolète de libtool dans l'archive. Comme elle ne prend pas en charge sysroot, les binaires produits seront liés aux bibliothèques de l'hôte par erreur. Contournez ce problème :

```
sed '6009s/$add_dir//' -i ltmain.sh
```

Créez de nouveau un répertoire de construction séparé :

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

Préparez la compilaton de Binutils :

```
../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 :

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

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

6.18. GCC-12.1.0 — Passe 2

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

Temps de construction approximatif: 14 SBU
Espace disque requis: 4.5 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
```

Override the building rule of libgcc and libstdc++ headers, to allow building these libraries with POSIX threads support:

```
sed '/thread_header =/s/@.*@/gthr-posix.h/' \
    -i libgcc/Makefile.in libstdc++-v3/include/Makefile.in
```

Créez de nouveau un répertoire de construction séparé :

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

Avant de commencer la construction de GCC, rappelez-vous d'effacer (avec **unset**) toute variable d'environnement surchargeant les options d'optimisation par défaut.

Maintenant, préparez la compilation de GCC :

```
../configure                                       \
    --build=$(../config.guess)                     \
    --host=$LFS_TGT                                \
    --target=$LFS_TGT                              \
    LDFLAGS_FOR_TARGET=-L$PWD/$LFS_TGT/libgcc      \
    --prefix=/usr                                  \
    --with-build-sysroot=$LFS                      \
    --enable-initfini-array                        \
    --disable-nls                                  \
    --disable-multilib                             \
    --disable-decimal-float                        \
    --disable-libatomic                            \
    --disable-libgomp                              \
    --disable-libquadmath                          \
    --disable-libssp                               \
    --disable-libvtv                               \
    --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.

`--target=$LFS_TGT`

As we are cross-compiling GCC, it's impossible to build target libraries (`libgcc` and `libstdc++`) with the compiled GCC binaries because these binaries won't run on the host distro. GCC building system will attempt to use the C and C++ compilers on the host distro as a workaround by default. It's not supported to build GCC target libraries with a different version of GCC, so using host compilers may cause building failure. This parameter ensures to build the libraries with GCC pass 1 and prevent the issue.

`LDFLAGS_FOR_TARGET=...`

Allow `libstdc++` to use shared `libgcc` being built in this pass, instead of the static version built in GCC pass 1. This is needed for supporting C++ exception handling.

`--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 sur ce paquet sont disponibles dans 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 comment construire les dernières parties manquantes du système temporaire : les outils requis par la machinerie de construction de plusieurs paquets. 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**.

Until Section 7.4, « Entrer dans l'environnement chroot », the commands must be run as `root`, with the `LFS` variable set. After entering chroot, all commands are run as `root`, fortunately without access to the OS of the computer you built LFS on. Be careful anyway, as it is easy to destroy the whole LFS system with badly formed commands.

7.2. Changement du propriétaire

Note

Les commandes dans le reste de ce livre doivent être exécutées lorsque vous êtes connecté en tant qu'utilisateur `root` et non en tant qu'utilisateur `lfs`. Revérifiez également que `$LFS` est paramétré dans l'environnement de `root`.

actuelement, la hiérarchie complète des répertoires de `$LFS` appartient à l'utilisateur `lfs`, un utilisateur qui n'existe que sur le système hôte. Si les répertoires et les fichiers dans `$LFS` sont conservés 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 de tous les fichiers du répertoire `$LFS`, ce qui exposerait alors ces fichiers à de possibles manipulations malveillantes.

Pour éviter ce problème, changez le propriétaire du répertoire `$LFS` pour l'attribuer 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. Monter et peupler /dev

During a normal boot, the kernel automatically mounts the `devtmpfs` filesystem on the `/dev` directory, and allow the devices to be created dynamically on that virtual filesystem as they are detected or accessed. Device creation is generally done during the boot process by the kernel and Udev. Since this new system does not yet have Udev

and has not yet been booted, it is necessary to mount and populate /dev manually. This is accomplished by bind mounting the host system's /dev directory. A bind mount is a special type of mount that allows you to create a mirror of a directory or mount point to some other location. Use the following command to achieve this:

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

7.3.2. Monter les systèmes de fichiers virtuels du noyau

Montez maintenant les systèmes de fichiers virtuels du noyau restants :

```
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 l'installation des outils temporaires restants. Nous utiliserons aussi cet environnement pour l'installation du système final. En tant que root, lancez la commande suivante pour entrer dans cet environnement qui, pour le moment, contient seulement les outils temporaires :

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

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 rétablis. La construction *TERM=$TERM* définira la variable TERM à l'intérieur du chroot avec la même valeur qu'à l'extérieur du chroot. Cette variable est nécessaire pour que des programmes comme **vim** et **less** fonctionnent correctement. Si vous avez besoin d'autres variables, telles que CFLAGS ou CXXFLAGS, c'est le bon endroit pour les rétablir.

À partir de maintenant, l'utilisation de la variable LFS n'est plus nécessaire parce que tout le travail sera restreint au système de fichiers LFS. Ceci s'explique par le fait que l'interpréteur de commande Bash apprend 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.

Remarquez que l'invite **bash** affichera 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 au sein du reste de ce chapitre et des chapitres suivants soient exécutées 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 du noyau sont montés comme expliqué dans Section 7.3.1, « Monter et peupler /dev » et Section 7.3.2, « Monter les systèmes de fichiers virtuels du noyau » et entrez de nouveau dans le chroot avant de continuer l'installation.

7.5. Création des répertoires

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

Note

Some of the directories mentioned in this section may be already created earlier with explicit instructions or when installing some packages. They are repeated below for completeness.

Créez quelques répertoires dans la racine qui ne font pas partie de l'ensemble limité requis dans les chapitres précédents, à l'aide de la commande suivante :

```
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/}{include,src}
mkdir -pv /usr/local/{bin,lib,sbin}
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 le mode d'autorisation 755, ce qui n'est pas souhaitable pour tous les répertoires. Dans la commande ci-dessus, deux modifications sont effectuées : une pour le répertoire principal de `root` et une autre pour les répertoires des fichiers temporaires.

Le premier changement de mode garantit que n'importe qui ne pourra pas entrer dans le répertoire `/root`, de façon identique à ce que ferait un utilisateur normal pour son répertoire principal. Le deuxième changement veille à ce que tout utilisateur puisse écrire dans les répertoires `/tmp` et `/var/tmp`, mais ne puisse 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*, standard de hiérarchies de fichiers, disponible sur *https://refspecs.linuxfoundation.org/fhs.shtml*). Le FHS mentionne 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. Création des fichiers et des liens symboliques essentiels

Historically, Linux maintains a list of the mounted file systems in the file `/etc/mtab`. Modern kernels maintain this list internally and expose it to the user via the `/proc` filesystem. To satisfy utilities that expect the presence of `/etc/mtab`, create the following symbolic link:

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

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

```
cat > /etc/hosts << EOF
127.0.0.1  localhost $(hostname)
::1        localhost
EOF
```

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:/usr/bin/false
daemon:x:6:6:Daemon User:/dev/null:/usr/bin/false
messagebus:x:18:18:D-Bus Message Daemon User:/run/dbus:/usr/bin/false
uuidd:x:80:80:UUID Generation Daemon User:/dev/null:/usr/bin/false
nobody:x:65534:65534:Unprivileged User:/dev/null:/usr/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:
input:x:24:
mail:x:34:
kvm:x:61:
uuidd:x:80:
wheel:x:97:
users:x:999:
nogroup:x:65534:
EOF
```

The created groups are not part of any standard—they are groups decided on in part by the requirements of the Udev configuration in Chapter 9, and in part by common convention employed by a number of existing Linux distributions. In addition, some test suites rely on specific users or groups. The Linux Standard Base (LSB, available at *http://refspecs.linuxfoundation.org/lsb.shtml*) only recommends that, besides the group `root` with a Group ID (GID) of 0, a group `bin` with a GID of 1 be present. The GID of 5 is widely used for `tty` group, and the number 5 is also used in `/etc/fstab` for the `devpts` filesystem. All other group names and GIDs can be chosen freely by the system administrator since well-written programs do not depend on GID numbers, but rather use the group's name.

The ID 65534 is used by the kernel for NFS and separate user namespaces for unmapped users and groups (those exist on the NFS server or the parent user namespace, but « do not exist » on the local machine or in the separate namespace). We assign `nobody` and `nogroup` for it to avoid an unnamed ID. But other distros may treat this ID differently, so any portable program should not depend on this assignment.

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

```
echo "tester:x:101:101::/home/tester:/bin/bash" >> /etc/passwd
echo "tester:x:101:" >> /etc/group
install -o tester -d /home/tester
```

Pour supprimer l'invite « I have no name! », démarrez un nouvel interpréteur de commandes. Puisque les fichiers `/etc/passwd` et `/etc/group` ont été créés, la résolution du nom d'utilisateur et du nom de groupe fonctionnera à présent :

```
exec /usr/bin/bash --login
```

Les programmes **login**, **agetty** et **init**, entre autres, utilisent un certain nombre de fichiers journaux pour enregistrer des informations qui permettent de savoir qui s'est connecté sur le système et quand. Cependant, ces programmes n'écriront pas vers ces fichiers journaux s'ils n'existent pas déjà. Initialisez les fichiers journaux et donnez-leur les droits nécessaires :

```
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 où chaque utilisateur s'est connecté pour la dernière fois. Le fichier `/var/log/faillog` enregistre les échecs de connexion. 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. 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: 1.6 SBU
Espace disque requis: 280 Mo

7.7.1. 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 de l'option de configuration :

`--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 sur ce paquet sont disponibles dans Section 8.31.2, « Contenu de Gettext. »

7.8. Bison-3.8.2

Le paquet Bison contient un générateur d'analyseurs.

Temps de construction approximatif: 0.3 SBU
Espace disque requis: 50 Mo

7.8.1. Installation de Bison

Préparez la compilation de Bison :

```
./configure --prefix=/usr \
            --docdir=/usr/share/doc/bison-3.8.2
```

Voici la signification des nouvelles options de configure :

`--docdir=/usr/share/doc/bison-3.8.2`
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 sur ce paquet sont disponibles dans Section 8.32.2, « Contenu de Bison. »

7.9. Perl-5.36.0

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

Temps de construction approximatif: 1.6 SBU
Espace disque requis: 272 Mo

7.9.1. Installation de Perl

Préparez la compilation de Perl :

```
sh Configure -des                                         \
             -Dprefix=/usr                                \
             -Dvendorprefix=/usr                          \
             -Dprivlib=/usr/lib/perl5/5.36/core_perl      \
             -Darchlib=/usr/lib/perl5/5.36/core_perl      \
             -Dsitelib=/usr/lib/perl5/5.36/site_perl      \
             -Dsitearch=/usr/lib/perl5/5.36/site_perl     \
             -Dvendorlib=/usr/lib/perl5/5.36/vendor_perl \
             -Dvendorarch=/usr/lib/perl5/5.36/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 sur ce paquet sont disponibles dans Section 8.41.2, « Contenu de Perl. »

7.10. Python-3.10.5

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: 1.2 SBU
Espace disque requis: 359 Mo

7.10.1. Installation de Python

Note

Il y a deux fichiers de paquet dont le nom commence par « python ». Celui à extraire est `Python-3.10.5.tar.xz` (attention à la majuscule sur la première lettre).

Préparez la compilation de Python :

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

Voici la signification de l'option de configuration :

`--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
```

Note

Certains modules Python 3 ne peuvent pas être construits à cause de dépendances qui ne sont pas encore installées. Le système de construction essaiera quand même de les construire, donc la compilation de certains fichiers échouera et les messages du compilateur indiqueront « fatal error ». Vous devriez ignorer ce message. Assurez-vous seulement que la commande **make** de plus haut niveau n'a pas échoué. Les modules facultatifs ne sont pas encore requis et ils seront construits dans le Chapitre 8.

Installez le paquet :

```
make install
```

Les détails sur ce paquet sont disponibles dans Section 8.50.2, « Contenu de Python 3. »

7.11. Texinfo-6.8

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

Temps de construction approximatif: 0.2 SBU
Espace disque requis: 109 Mo

7.11.1. Installation de Texinfo

Tout d'abord, corrigez un problème de construction du paquet avec Glibc-2.34 et ultérieur :

```
sed -e 's/__attribute_nonnull__/__nonnull/' \
    -i gnulib/lib/malloc/dynarray-skeleton.c
```

Préparez la compilation de Texinfo :

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

Compilez le paquet :

```
make
```

Installez le paquet :

```
make install
```

Les détails sur ce paquet sont disponibles dans Section 8.68.2, « Contenu de Texinfo. »

7.12. Util-linux-2.38

le paquet Util-linux contient divers programmes utilitaires.

Temps de construction approximatif: 0.7 SBU
Espace disque requis: 129 Mo

7.12.1. 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    \
            --libdir=/usr/lib    \
            --docdir=/usr/share/doc/util-linux-2.38 \
            --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 configuration :

`ADJTIME_PATH=/var/lib/hwclock/adjtime`
This sets the location of the file recording information about the hardware clock in accordance to the FHS. This is not strictly needed for this temporary tool, but it prevents creating a file at another location, which would not be overwritten or removed when building the final util-linux package.

`--libdir=/usr/lib`
Ce paramètre s'assure que les liens symboliques `.so` ciblent le fichier de la bibliothèque partagée directement dans le même répertoire (`/usr/lib`).

`--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 sur ce paquet sont disponibles dans Section 8.73.2, « Contenu d'Util-linux. »

7.13. Nettoyage et Sauvegarde du système temporaire

7.13.1. Nettoyage

Tout d'abord, supprimez la documentation actuellement installée pour éviter qu'elle ne se retrouve sur le système final, et pour récupérer environ 35 Mo :

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

Ensuite, les fichiers .la de libtool ne sont utiles que pour se lier à des bibliothèques statiques. Ils ne sont pas utiles voir potentiellement dangereux quand on utilise des bibliothèques partagées, surtout pour les systèmes de construction qui ne sont pas basés sur les autotools. Toujours dans le chroot, supprimez ces fichiers maintenant :

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

La taille du système est maintenant d'environ 3 Go, mais le répertoire /tools n'est plus requis. Il utilise environ 1 Go d'espace disque. Supprimez-le maintenant :

```
rm -rf /tools
```

7.13.2. Sauvegarde

Maintenant, les programmes et bibliothèques essentiels ont été créés et votre système LFS actuel est en bon état. Votre système peut maintenant être sauvegardé pour être réutilisé plus tard. Si vous rencontrez une erreur fatale dans les chapitres suivants, il arrive souvent que tout supprimer et recommencer (avec plus de prudence) soit la meilleure option. Malheureusement, tous les fichiers temporaires seront aussi supprimés. Pour éviter de passer du temps en plus pour refaire quelque chose que vous avez déjà réussi, préparez une sauvegarde.

Note

Toutes les étapes restantes dans cette section sont facultatives. Cependant, dès que vous commencez à installer des paquets dans le Chapitre 8, les fichiers temporaires seront remplacés. Donc c'est peut-être une bonne idée d'effectuer une sauvegarde du système actuel, comme on le décrit plus bas.

Les étapes suivantes sont à effectuer en dehors de l'environnement chroot. Cela signifie que vous devez quitter l'environnement chroot avant de continuer. La raison est est qu'il faut pouvoir accéder à des emplacement du système de fichiers en dehors de l'environnement chroot pour stocker et lire l'archive de sauvegarde, qui ne devrait pas se trouver dans la hiérarchie `$LFS` pour plus de sûreté.

Si vous avez décidé d'effectuer une sauvegarde, quittez l'environnement chroot :

```
exit
```

Important

Toutes les instructions suivantes sont exécutées en `root` sur votre système hôte. Faites particulièrement attention aux commandes que vous allez exécuter car toute erreur ici peut modifier votre environnement hôte. Soyez conscient que la variable d'environnement `LFS` a une valeur pour l'utilisateur `lfs` par défaut, mais peut ne *pas* exister pour `root`.

Quand les commandes doivent être exécutées par `root`, assurez-vous d'avoir la variable `LFS`.

On en a déjà parlé dans le Section 2.6, « Définition de la variable $LFS ».

Avant de faire la sauvegarde, démontez les systèmes de fichiers virtuels :

```
umount $LFS/dev/pts
umount $LFS/{sys,proc,run,dev}
```

Assurez-vous d'avoir au moins 1 Go d'espace disque libre (les archives des sources seront incluses dans l'archive de sauvegarde) sur le système de fichier contenant le répertoire dans lequel vous créez la sauvegarde.

Remarquez que les instructions ci-dessous spécifient le répertoire personnel de l'utilisateur `root` sur le système hôte, qui se trouve généralement sur le système de fichiers racine.

Remplacez `$HOME` par un répertoire de votre choix si vous ne voulez pas stoker la sauvegarde dans le répertoire personnel de `root`.

Créez l'archive de sauvegarde en lançant la commande suivante :

Note

Because the backup archive is compressed, it takes a relatively long time (over 10 minutes) even on a reasonably fast system.

```
cd $LFS
tar -cJpf $HOME/lfs-temp-tools-r11.1-135+.tar.xz .
```

Note

Si vous continuez au chapitre 8, n'oubliez pas d'entrer de nouveau dans l'environnement chroot comme expliqué dans l'encadré « important » plus bas.

7.13.3. Restauration du système

Dans le cas où vous avez fait des erreurs et que vous devez recommencer du début, vous pouvez utiliser cette sauvegarde pour réinitialiser le système et gagner du temps. Comme les sources se trouvent dans `$LFS`, elles sont incluses dans l'archive de sauvegarde, donc vous n'aurez pas besoin de les télécharger de nouveau. Après avoir vérifié que `$LFS` est définie correctement, restaurez la sauvegarde en exécutant les commandes suivantes :

Avertissement

The following commands are extremely dangerous. If you run **rm -rf ./*** as the `root` user and you do not change to the $LFS directory or the `LFS` environment variable is not set for the `root` user, it will destroy your entire host system. YOU ARE WARNED.

```
cd $LFS
rm -rf ./*
tar -xpf $HOME/lfs-temp-tools-r11.1-135+.tar.xz
```

De nouveau, vérifiez que l'environnement a été correctement paramétré et continuez à construire le reste du système.

Important

Si vous quittez l'environnement chroot pour créer une sauvegarde ou pour recommencer à construire à partir d'une sauvegarde, rappelez-vous de vérifier que les systèmes de fichiers virtuels sont toujours montés (**findmnt | grep $LFS**). S'ils ne sont pas montés, remontez-les maintenant comme décrit dans le Section 7.3, « Préparer les systèmes de fichiers virtuels du noyau » et entrez de nouveau dans l'environnement chroot (voir le 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 le noyau Linux doit être mis à jour (par exemple de 5.10.17 à 5.10.18 ou 5.11.1), rien d'autre n'a besoin d'être reconstruit. Le système continuera de fonctionner grâce à l'interface bien définie entre le noyau et l'espace utilisateur. En particulier, les en-têtes de l'API de Linux n'ont pas besoin d'être mises à jour (et ne devraient pas l'être, voir le point suivant) avec le noyau. Vous devrez redémarrer votre système pour utiliser le noyau à jour.
- Si les en-têtes de l'API Linux doivent être mis à jour vers une nouvelle version, 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.
- Si vous mettez à jour un paquet qui contient une bibliothèque partagée, et que le nom de la bibliothèque ne change pas, mais que le numéro de version du **fichier** de la bibliothèque décroît (par exemple le nom reste `libfoo.so.1`, mais le nom du fichier de la bibliothèque change de `libfoo.so.1.25` à `libfoo.so.1.24`), vous devrez supprimer le fichier de bibliothèque de la version précédente (`libfoo.so.1.25` dans ce cas). Sinon, en lançant **ldconfig** (par vous-même via la ligne de commande, ou par l'installation d'un paquet) vous réinitialiserez le lien symbolique `libfoo.so.1` vers l'ancien fichier de bibliothèque parce qu'il a une version « plus récente », puisque le numéro est plus grand. Cette situation arrive quand vous installez une version précédente d'un paquet, ou que le paquet change de pratique de nommage des versions.
- Si un paquet contenant une bibliothèque partagée est mis à jour, et que le nom de la bibliothèque ne change pas, mais qu'un problème important (en particulier une vulnérabilité de sécurité) est corrigé, tous les programmes en cours d'exécution liés à la bibliothèque partagée doivent être redémarrés. La commande suivante, lancée en `root` après la mise à jour, affichera ce qui utilise les anciennes versions de ces bibliothèques (remplacez *`libfoo`* par le nom de la bibliothèque) :

```
grep -l  -e 'libfoo.*deleted' /proc/*/maps |
   tr -cd 0-9\\n | xargs -r ps u
```

Si OpenSSH est utilisé pour accéder au système et qu'il est lié à la bibliothèque mise à jour, vous devrez redémarrer le service **sshd**, vous déconnecter, vous reconnecter et relancer la commande pour confirmer que plus rien n'utilise encore les bibliothèques supprimées.

- Si un binaire ou une bibliothèque partagée est remplacé, les processus utilisant le code ou les données du binaire ou de la bibliothèque peuvent crasher. La manière correcte de mettre à jour un binaire ou une bibliothèque partagée sans causer de crash de processus est de le ou la supprimer d'abord, puis d'installer la nouvelle version. La commande **install** fournie par Coreutils implémente déjà cela et la plupart des paquets l'utilisent pour installer des binaires et des bibliothèques. Cela signifie que vous n'aurez pas ce problème la plupart du temps. Cependant, le processus d'installation de certains paquets (notamment Mozilla JS dans BLFS) se contentent de réécrire par dessus le fichier s'il existe déjà et cause un crash, donc il est plus prudent de sauvegarder votre travail et de fermer les processus inutiles avant de mettre à jour un paquet.

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`, `/etc/ld.so.conf`, `/etc/sysconfig/rc.site`, `/etc/sysconfig/network` et `/etc/sysconfig/ifconfig.eth0`.

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-5.13

Le paquet Man-pages contient environ 2 200 pages de manuel.

Temps de construction approximatif: moins de 0.1 SBU
Espace disque requis: 33 Mo

8.3.1. Installation de Man-pages

Installez Man-pages en lançant :

```
make prefix=/usr install
```

8.3.2. Contenu de Man-pages

Fichiers installés: différentes pages de manuel

Descriptions courtes

`pages man` Décrivent les fonctions du langage de programmation C, les fichiers périphériques et les fichiers de configuration importants

8.4. Iana-Etc-20220524

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-2.35

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: 24 SBU
Espace disque requis: 2.8 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-2.35-fhs-1.patch
```

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

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

Ensure that the **ldconfig** and **sln** utilities will be installed into `/usr/sbin`:

```
echo "rootsbindir=/usr/sbin" > configparms
```

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=/usr/lib
```

Voici la signification des options de configuration :

`--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=/usr/lib`
Cette variable indique la bibliothèque correcte pour chaque système. Nous ne voulons pas utiliser lib64.

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. Vous pouvez généralement ignorer quelques échecs parmi les plus de 4200 tests. 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.
- The *nss/tst-nss-files-hosts-long* test is known to fail if the system has no non-loopback IP addresses.
- The *nss/tst-nss-files-hosts-multi* test is known to fail on relatively slow systems due to an internal timeout.

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
```

Corrigez le chemin codé en dur vers le chargeur d'exécutable dans le script **ldd** :

```
sed '/RTLDLIST=/s@/usr@@g' -i /usr/bin/ldd
```

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
```

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 seconde commande **localedef** ci-dessous combine la définition de la locale indépendante du codage `/usr/share/i18n/locales/cs_CZ` avec la définition de la page de codes `/usr/share/i18n/charmaps/UTF-8.gz` et ajoute le résultat à la fin du 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 ISO-8859-1 en_GB
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_ES -f ISO-8859-15 es_ES@euro
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 is_IS -f ISO-8859-1 is_IS
localedef -i is_IS -f UTF-8 is_IS.UTF-8
localedef -i it_IT -f ISO-8859-1 it_IT
localedef -i it_IT -f ISO-8859-15 it_IT@euro
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.SJIS 2> /dev/null || true
localedef -i ja_JP -f UTF-8 ja_JP.UTF-8
localedef -i nl_NL@euro -f ISO-8859-15 nl_NL@euro
localedef -i ru_RU -f KOI8-R ru_RU.KOI8-R
localedef -i ru_RU -f UTF-8 ru_RU.UTF-8
localedef -i se_NO -f UTF-8 se_NO.UTF-8
localedef -i ta_IN -f UTF-8 ta_IN.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
localedef -i zh_TW -f UTF-8 zh_TW.UTF-8
```

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-2.35/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-2.35/localedata/SUPPORTED` dans le cas peu probable où vous en auriez besoin. Par exemple, les deux paramètres linguistiques suivants sont requis par certains tests plus tard dans ce chapitre :

```
localedef -i POSIX -f UTF-8 C.UTF-8 2> /dev/null || true
localedef -i ja_JP -f SHIFT_JIS ja_JP.SJIS 2> /dev/null || true
```

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 ../../tzdata2022a.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: gencat, getconf, getent, iconv, iconvconfig, ldconfig, ldd, lddlibc4, ld.so (lien symbolique vers ld-linux-x86-64.so.2 ou ld-linux.so.2), locale, localedef, makedb, mtrace, nscd, pcprofiledump, pldd, sln, sotruss, sprof, tzselect, xtrace, zdump et zic

Bibliothèques installées: ld-linux-x86-64.so.2, ld-linux.so.2, libBrokenLocale.{a,so}, libanl.{a,so}, libc.{a,so}, libc_nonshared.a, libc_malloc_debug.so, libcrypt.{a,so}, libdl.{a,so.2}, libg.a, libm.{a,so}, libmcheck.a, libmemusage.so, libmvec.{a,so}, libnsl.so.1, libnss_compat.so, libnss_dns.so, libnss_files.so, libnss_hesiod.so, libpcprofile.so, libpthread.{a,so.0}, libresolv.{a,so}, librt.{a,so.1}, libthread_db.so et libutil.{a,so.1}

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

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. Elle n'existe pas sur les nouvelles architectures comme x86_64
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-*.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-2.35/locale/broken_cur_max.c` pour plus d'informations
`libanl`	Une bibliothèque asynchrone de recherche de noms
`libc`	La principale bibliothèque C
`libc_malloc_debug`	Active le test d'allocation de mémoire lorsqu'elle est préchargée
`libcrypt`	La bibliothèque de chiffrement
`libdl`	Bibliothèque factice ne contenant aucune fonction. C'était auparavant une bibliothèque d'interface de liaison dynamique, dont les fonctions sont maintenant dans `libc`
`libg`	Bibliothèque factice ne contenant aucune fonction. C'était auparavant une bibliothèque d'exécution pour **g++**

`libm`	La bibliothèque mathématique
`libmvec`	La bibliothèque de mathématique vectorielle, liée si besoin quand `libm` est utilisée
`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, maintenant obsolète
`libnss_*`	Les modules du 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. Chargée par la `libc` en fonction de la configuration présente dans le fichier `/etc/nsswitch.conf`
`libpcprofile`	Peut être préchargé pour profiler le PC d'un exécutable
`libpthread`	Bibliothèque factice ne contenant aucune fonction. Elle contenait auparavant les fonctions fournissant la plupart des interfaces spécifiées par l'extension temps réel de POSIX.1b, maintenant ces fonctions sont dans `libc`
`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`	Bibliothèque factice ne contenant aucune fonction. Elle contenait auparavant du code pour les fonctions « standard » utilisées dans de nombreux utilitaires Unix différents. Ces fonctions sont maintenant dans `libc`

8.6. Zlib-1.2.12

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
```

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 de 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: 7.2 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`
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 les bibliothèques partagées :

```
cp -av libbz2.so.* /usr/lib
ln -sv libbz2.so.1.0.8 /usr/lib/libbz2.so
```

Installez le binaire partagé **bzip2** dans le répertoire `/usr/bin`, et remplacez deux copies de **bzip2** par des liens symboliques :

```
cp -v bzip2-shared /usr/bin/bzip2
for i in /usr/bin/{bzcat,bunzip2}; do
  ln -sfv bzip2 $i
done
```

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

Apply a patch to fix an issue in xzgrep:

```
patch -Np1 -i ../xz-5.2.5-upstream_fix-1.patch
```

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 :

```
make install
```

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-1.5.2

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: 55 Mo

8.9.1. Installation de Zstd

Apply a patch to fix some issues identified by upstream:

```
patch -Np1 -i ../zstd-1.5.2-upstream_fixes-1.patch
```

Compilez le paquet :

```
make prefix=/usr
```

Note

Il sera indiqué « failed » à plusieurs endroits dans la sortie des tests. C'est attendu et seul « FAIL » est un vrai échec des tests. Il ne devrait pas y avoir d'échec.

Pour tester les résultats, lancez :

```
make check
```

Installez le paquet :

```
make prefix=/usr install
```

Supprimez la bibliothèque statique :

```
rm -v /usr/lib/libzstd.a
```

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-5.42

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

Temps de construction approximatif: 0.1 SBU
Espace disque requis: 15 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 grâce à l'exécution de 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 que le programme **file** utilise

8.11. Readline-8.1.2

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: 15 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-8.1.2
```

Voici la signification de l'option de configuration :

`--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
```

Si désiré, installez la documentation :

```
install -v -m644 doc/*.{ps,pdf,html,dvi} /usr/share/doc/readline-8.1.2
```

8.11.2. Contenu de Readline

Bibliothèques installées: libhistory.so et libreadline.so
Répertoires installés: /usr/include/readline et /usr/share/doc/readline-8.1.2

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. M4-1.4.19

Le paquet M4 contient un processeur de macros.

Temps de construction approximatif: 0.7 SBU
Espace disque requis: 49 Mo

8.12.1. Installation de M4

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-5.3.1

Le paquet Bc contient un langage de traitement des nombres en précision arbitraire.

Temps de construction approximatif: moins de 0.1 SBU
Espace disque requis: 7.1 Mo

8.13.1. Installation de Bc

Préparez la compilation de Bc :

```
CC=gcc ./configure --prefix=/usr -G -O3
```

Voici la signification des options de configuration :

CC=gcc
Ce paramètre spécifie le compilateur à utiliser.

-O3
Spécifie le niveau d'optimisation à utiliser.

-G
élimine 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 notation polonaise inverse

8.14. Flex-2.6.4

Le paquet Flex contient un outil de génération de programmes qui reconnaissent des motifs dans dans du texte.

Temps de construction approximatif: 0.4 SBU
Espace disque requis: 32 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 aider ces programmes, créez un lien symbolique nommé `lex` qui lance `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 qui reconnaissent des motifs dans un texte. Cela permet une grande adaptabilité lors du choix des règles de recherche de motifs, ce qui élimine ainsi le besoin de développer un programme spécialisé

flex++ Une extension de flex 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-8.6.12

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: 3.4 SBU
Espace disque requis: 87 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 ../tcl8.6.12-html.tar.gz --strip-components=1
```

Préparez la compilation de Tcl :

```
SRCDIR=$(pwd)
cd unix
./configure --prefix=/usr           \
            --mandir=/usr/share/man
```

Construisez le paquet :

```
make

sed -e "s|$SRCDIR/unix|/usr/lib|" \
    -e "s|$SRCDIR|/usr/include|"  \
    -i tclConfig.sh

sed -e "s|$SRCDIR/unix/pkgs/tdbc1.1.3|/usr/lib/tdbc1.1.3|" \
    -e "s|$SRCDIR/pkgs/tdbc1.1.3/generic|/usr/include|"    \
    -e "s|$SRCDIR/pkgs/tdbc1.1.3/library|/usr/lib/tcl8.6|" \
    -e "s|$SRCDIR/pkgs/tdbc1.1.3|/usr/include|"            \
    -i pkgs/tdbc1.1.3/tdbcConfig.sh

sed -e "s|$SRCDIR/unix/pkgs/itcl4.2.2|/usr/lib/itcl4.2.2|" \
    -e "s|$SRCDIR/pkgs/itcl4.2.2/generic|/usr/include|"    \
    -e "s|$SRCDIR/pkgs/itcl4.2.2|/usr/include|"            \
    -i pkgs/itcl4.2.2/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
```

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
```

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
```

Si vous avez téléchargé la documentation facultative, installez-la en lançant la commande suivante :

```
mkdir -v -p /usr/share/doc/tcl-8.6.12
cp -v -r  ../html/* /usr/share/doc/tcl-8.6.12
```

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 tâches qui sont trop compliquées avec quoi que ce soit d'autre. Le cadre de tests DejaGnu est écrit en Expect.

Temps de construction approximatif: 0.2 SBU
Espace disque requis: 3.9 Mo

8.16.1. Installation d'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 configuration :

`--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: libexpect5.45.4.so

Descriptions courtes

expect Communique avec les autres programmes interactifs selon un script.

`libexpect-5.45.4.so` Contient des fonctions qui permettent à Expect d'être utilisé comme une extension Tcl ou directement à partir du langage C ou du langage C++ (sans Tcl)

8.17. DejaGNU-1.6.3

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: 6.9 Mo

8.17.1. Installation de DejaGNU

Les développeurs en amont recommandent de construire DejaGNU dans un répertoire de construction dédié :

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

Préparez la compilation de DejaGNU :

```
../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-1.6.3
install -v -m644   doc/dejagnu.{html,txt} /usr/share/doc/dejagnu-1.6.3
```

Pour tester les résultats, lancez :

```
make check
```

8.17.2. Contenu de DejaGNU

Programme installé: dejagnu et runtest

Descriptions courtes

dejagnu Lanceur de commande auxiliaire de DejaGNU

runtest Un script enveloppe qui repère le bon shell **expect** puis lance DejaGNU

8.18. Binutils-2.38

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

Temps de construction approximatif: 6.1 SBU
Espace disque requis: 4.6 Go

8.18.1. Installation de Binutils

Vérifiez que les pseudo-terminaux (PTY) fonctionnent correctement dans l'environnement chroot 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.
```

Les développeurs amont ont créé un correctif pour corriger un problème lors de l'utilisation de binutils pour construire certains paquets BLFS qui ont activé l'optimisation à l'édition des liens (LTO). Appliquez-le maintenant :

```
patch -Np1 -i ../binutils-2.38-lto_fix-1.patch
```

Maintenant appliquez un correctif identifié en amont qui affecte la construction de certains paquets :

```
sed -e '/R_386_TLS_LE /i \   || (TYPE) == R_386_TLS_IE \\' \
    -i ./bfd/elfxx-x86.h
```

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 la fois en tant que ld (l'éditeur par défaut) et que 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 inférieure). 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 pour être `$(exec_prefix)/$(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 compilation croisée (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
```

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 associés à cette adresse
ar	Crée, modifie et extrait à partir d'archives
as	Un assembleur qui assemble la sortie de **gcc** en fichiers objets
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 du graphe d'appels
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 matériel vers **ld**
nm	Liste les symboles présents dans un fichier objet
objcopy	Traduit un type de fichier objet en un autre
objdump	Affiche des informations sur le fichier objet donné, avec des options contrôlant les informations à afficher. Les informations affichées sont surtout utiles 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, uniquement 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 d'arithmétique multi-précision.

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 dans cette section 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 arithmétique multi-précision

8.21. MPC-1.2.1

Le paquet MPC contient une bibliothèque pour le calcul arithmétique avec des nombres complexes à précision arbitraire et l'arrondi correct du résultat.

Temps de construction approximatif: 0.3 SBU
Espace disque requis: 21 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 et générez la documentation HTML :

```
make
make html
```

Pour tester les résultats, lancez :

```
make check
```

Installez le paquet ainsi que sa documentation :

```
make install
make install-html
```

8.21.2. Contenu de MPC

Bibliothèques installées: libmpc.so
Dossier installé: /usr/share/doc/mpc-1.2.1

Descriptions courtes

`libmpc` Contient des fonctions mathématiques complexes

8.22. Attr-2.5.1

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: 4.1 Mo

8.22.1. Installation d'Attr

Préparez la compilation d'Attr :

```
./configure --prefix=/usr     \
            --disable-static  \
            --sysconfdir=/etc \
            --docdir=/usr/share/doc/attr-2.5.1
```

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
```

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-2.5.1

Descriptions courtes

attr Étend les attributs 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 fonctions pour la manipulation des attributs étendus

8.23. Acl-2.3.1

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: 6.1 Mo

8.23.1. Installation d'Acl

Préparez la compilation d'Acl :

```
./configure --prefix=/usr         \
            --disable-static      \
            --docdir=/usr/share/doc/acl-2.3.1
```

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
```

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-2.3.1

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-2.64

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: 2.7 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 :

```
make prefix=/usr lib=lib install
```

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 capacités
getcap Examine les capacités d'un fichier
getpcaps Affiche les capacités d'un ou plusieurs processus
setcap Définit les capacités d'un fichier
`libcap` Contient les fonctions de la bibliothèque de manipulation des capacités 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-4.11.1

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: 49 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-svn/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-5.13 » :

```
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'emplacement 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. Ensuite, retirez `/bin` et `/sbin` de `PATH`, car ce sont de simples liens symboliques vers leur contrepartie dans `/usr`.

Note

Si vous préférez garder `/bin` ou `/sbin` dans `PATH` pour une raison ou une autre, modifiez `PATH` dans `.bashrc` après la construction de LFS.

```
sed -e 's:#ENCRYPT_METHOD DES:ENCRYPT_METHOD SHA512:' \
    -e 's:/var/spool/mail:/var/mail:'                 \
    -e '/PATH=/{s@/sbin:@@;s@/bin:@@}'                \
    -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
```

Préparez la compilation de Shadow :

```
touch /usr/bin/passwd
./configure --sysconfdir=/etc \
            --disable-static  \
            --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 exec_prefix=/usr install
make -C man install-man
```

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 par défaut de 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`* ou *`-N`* à **useradd** ou changer le paramètre *`USERGROUPS_ENAB`* dans `/etc/login.defs`. Voir `useradd(8)` pour plus d'informations.

Deuxièmement, pour changer les paramètres par défaut, vous devez créer le fichier `/etc/default/useradd` et l'adapter à vos besoins. Créez-le avec :

```
mkdir -p /etc/default
useradd -D --gid 999
```

Explication des paramètres de `/etc/default/useradd`

`GROUP=999`
Ce paramètre initialise le début des numéros de groupe utilisés dans le fichier `/etc/group`. La valeur 999 particulière provient du paramètre *`--gid`* ci-dessus. 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 de GID égal à ce numéro 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 999`, bien que le compte soit correctement créé. C'est pourquoi nous avons créé le groupe `users` avec cet identifiant de groupe dans le Section 7.6, « Création des fichiers et des liens symboliques essentiels ».

`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 '/MAIL/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, getsubids, 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
Bibliothèques installées: libsubid.so

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

getsubids Est utilisé pour lister les intervalles des identifiants mineurs d'un utilisateurs

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`

`libsubid` bibliothèque de traitement des intervalles subordonnés d'id des utilisateurs

8.26. GCC-12.1.0

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

Temps de construction approximatif: 160 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 de programmation, il existe des prérequis qui ne sont pas encore disponibles. Consultez 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 nouveaux paramètres de configuration :

`LD=ld`
Ce paramètre permet de s'assurer que le script configure utilise le ld installé par binutils, construit plus tôt dans ce chapitre, au lieu de la version compilée de manière croisée qui serait autrement utilisée.

`--with-system-zlib`
Ce paramètre dit à GCC de se lier à la copie de la bibliothèque zlib installée sur le système, plutôt qu'à sa propre copie interne.

Compilez le paquet :

```
make
```

Important

Dans cette section, la suite de tests de GCC est considérée comme importante, mais elle prend beaucoup de temps. Les novices sont encouragés à ne pas l'ignorer. La durée des tests peut être significativement réduite en ajoutant -jx à la commande make ci-dessous, où x est le nombre de cœurs sur votre système.

Un ensemble de tests dans la suite de tests de GCC est connu pour utiliser toute la pile par défaut, c'est pourquoi vous devez augmenter 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 *https://www.linuxfromscratch.org/lfs/build-logs/development/* et *https://gcc.gnu.org/ml/gcc-testresults/*.

In g++, four tests related to PR100400 are known to be reported as both XPASS and FAIL. It's because the test file for this known issue is not well written.

Quelques échecs inattendus sont parfois inévitables. Les développeurs de GCC connaissent généralement ces problèmes, mais ils ne les ont pas encore résolus. Sauf si les résultats des tests sont très différents de ceux sur l'adresse ci-dessus, vous pouvez poursuivre en toute sécurité.

Installez le paquet :

```
make install
```

Le répertoire de construction de GCC appartient maintenant à `tester` et la propriété du répertoire des en-têtes installé (et son contenu) sera incorrecte. Transférez la propriété à l'utilisateur et au groupe `root` :

```
chown -v -R root:root \
    /usr/lib/gcc/$(gcc -dumpmachine)/12.1.0/include{,-fixed}
```

Créez un lien symbolique requis par le *FHS* pour des raisons « historiques ».

```
ln -svr /usr/bin/cpp /usr/lib
```

Ajoutez un lien symbolique de compatibilité pour permettre la compilation de programmes avec l'optimisation à l'édition des liens (LTO) :

```
ln -sfv ../../libexec/gcc/$(gcc -dumpmachine)/12.1.0/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. Vous devez alors effectuer plusieurs contrôles d'intégrité :

```
echo 'int main(){}' > dummy.c
cc dummy.c -v -Wl,--verbose &> dummy.log
readelf -l a.out | grep ': /lib'
```

There should be no errors, and the output of the last command will be (allowing for platform-specific differences in the dynamic linker name):

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

Now make sure that we're setup to use the correct start files:

```
grep -o '/usr/lib.*/crt[1in].*succeeded' dummy.log
```

The output of the last command should be:

```
/usr/lib/gcc/x86_64-pc-linux-gnu/12.1.0/../../../../lib/crt1.o succeeded
/usr/lib/gcc/x86_64-pc-linux-gnu/12.1.0/../../../../lib/crti.o succeeded
/usr/lib/gcc/x86_64-pc-linux-gnu/12.1.0/../../../../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`. Il est important de vérifier que **gcc** a trouvé les trois fichiers `crt*.o` sous le répertoire `/usr/lib`.

Verify that the compiler is searching for the correct header files:

```
grep -B4 '^ /usr/include' dummy.log
```

This command should return the following output:

```
#include <...> search starts here:
 /usr/lib/gcc/x86_64-pc-linux-gnu/12.1.0/include
 /usr/local/include
 /usr/lib/gcc/x86_64-pc-linux-gnu/12.1.0/include-fixed
 /usr/include
```

À nouveau, le répertoire nommé selon votre triplet cible peut être différent de celui ci-dessus, selon l'architecture de votre système.

Next, verify that the new linker is being used with the correct search paths:

```
grep 'SEARCH.*/usr/lib' dummy.log |sed 's|; |\n|g'
```

References to paths that have components with '-linux-gnu' should be ignored, but otherwise the output of the last command should be:

```
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");
```

Un système 32 bits peut voir quelques répertoires différemment. 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");
```

Next make sure that we're using the correct libc:

```
grep "/lib.*/libc.so.6 " dummy.log
```

The output of the last command should be:

```
attempt to open /usr/lib/libc.so.6 succeeded
```

Make sure GCC is using the correct dynamic linker:

```
grep found dummy.log
```

The output of the last command should be (allowing for platform-specific differences in dynamic linker name):

```
found ld-linux-x86-64.so.2 at /usr/lib/ld-linux-x86-64.so.2
```

If the output does not appear as shown above or is not received at all, then something is seriously wrong. Investigate and retrace the steps to find out where the problem is and correct it. Any issues will need to be resolved before continuing with the process.

Once everything is working correctly, clean up the test files:

```
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, gcov-tool et lto-dump

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-12.1.0

Descriptions courtes

c++ Le compilateur C++

cc Le compilateur C

cpp Le préprocesseur C 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 à l'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, qui 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

lto-dump Outil pour afficher les fichiers objets produits par GCC quand LTO est activé

`libasan` La bibliothèque de vérification des adresses à l'exécution

`libatomic` Bibliothèque d'exécution intégrée pour les opérations atomiques de GCC

`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 si GCC active le profilage

`libgomp` L'implémentation GNU de l'API OpenMP API pour la programmation en mémoire parallèle partagée sur plusieurs plateformes en C/C++ et Fortran

`libitm` La bibliothèque mémoire transactionnelle de GNU

`liblsan` La bibliothèque de vérification de fuites à l'exécution

`liblto_plugin`	Le greffon LTO de GCC permet à binutils de traiter les fichiers objets créés par GCC quand LTO est activée
`libquadmath`	API de la bibliothèque mathématique en quadruple précision de GCC
`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 du langage de programmation C++
`libtsan`	La bibliothèque de vérification des threads à l'exécution
`libubsan`	La bibliothèque de vérification des comportements non définis 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: 29 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 matériel 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-6.3

Le paquet Ncurses contient les bibliothèques pour gérer les écrans type caractère indépendamment des terminaux.

Temps de construction approximatif: 0.4 SBU
Espace disque requis: 45 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          \
            --with-pkg-config-libdir=/usr/lib/pkgconfig
```

Voici la signification des nouvelles options de configuration :

`--enable-widec`
Cette option amène les bibliothèques « wide-character » (comme `libncursesw.so.6.3`) à être compilées au lieu des bibliothèques normales (comme `libncurses.so.6.3`). 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`
Cette option 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.

L'installation de ce paquet écrasera `libncursesw.so.6.3`. Cela peut faire crasher le processus de shell qui utilise du code et des données du fichier de bibliothèque. Installez le paquet avec `DESTDIR`, et remplacez le fichier de bibliothèque correctement avec la commande **install**. Une archive statique inutile qui n'est pas gérée par **configure** est aussi supprimée :

```
make DESTDIR=$PWD/dest install
install -vm755 dest/usr/lib/libncursesw.so.6.3 /usr/lib
rm -v  dest/usr/lib/{libncursesw.so.6.3,libncurses++w.a}
cp -av dest/* /
```

Beaucoup d'applications s'attendent encore à ce que l'éditeur de liens puisse trouver les bibliothèques Ncurses non wide-character. Faites en sorte que ces applications croient au lien vers les bibliothèques wide-character par des 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
```

Si désiré, installez la documentation de Ncurses :

```
mkdir -pv      /usr/share/doc/ncurses-6.3
cp -v -R doc/* /usr/share/doc/ncurses-6.3
```

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 connues qui se lient aux bibliothèques 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-6.3

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 qui traduit 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 donné

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 complet

tset	Peut être utilisé pour initialiser des terminaux
`libcursesw`	Un lien vers `libncursesw`
`libncursesw`	Contient des fonctions pour afficher du texte de plusieurs façons complexes 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 formulaires
`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: 0.4 SBU
Espace disque requis: 31 Mo

8.29.1. Installation de Sed

Préparez la compilation de Sed :

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

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-23.5

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: 5.6 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
```

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: 2.7 SBU
Espace disque requis: 233 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-3.8.2

Le paquet Bison contient un générateur d'analyseurs.

Temps de construction approximatif: 6.3 SBU
Espace disque requis: 53 Mo

8.32.1. Installation de Bison

Préparez la compilation de Bison :

```
./configure --prefix=/usr --docdir=/usr/share/doc/bison-3.8.2
```

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 en a besoin

8.33. Grep-3.7

Le paquet Grep contient des programmes de recherche du contenu de fichiers.

Temps de construction approximatif: 0.9 SBU
Espace disque requis: 36 Mo

8.33.1. Installation de Grep

Préparez la compilation de Grep :

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

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 qui correspondent à une expression rationnelle étendue

fgrep Affiche les lignes qui correspondent à une liste de chaînes fixes

grep Affiche des lignes qui correspondent à une expression rationnelle basique

8.34. Bash-5.1.16

Le paquet Bash contient le shell Bourne-Again.

Temps de construction approximatif: 1.5 SBU
Espace disque requis: 50 Mo

8.34.1. Installation de Bash

Préparez la compilation de Bash :

```
./configure --prefix=/usr                      \
            --docdir=/usr/share/doc/bash-5.1.16 \
            --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` déjà installée 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 .
```

The testsuite of the package is designed to be run as a non-`root` user that owns the terminal connected to standard input. To satisfy the requirement, spawn a new pseudo terminal using Expect and run the tests as the `tester` user:

```
su -s /usr/bin/expect tester << EOF
set timeout -1
spawn make tests
expect eof
lassign [wait] _ _ _ value
exit $value
EOF
```

Installez le paquet :

```
make install
```

Lancez le programme **bash** nouvellement compilé (en remplaçant celui en cours d'exécution) :

```
exec /usr/bin/bash --login
```

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-5.1.16

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, ce qui fait de cet interpréteur un outil très puissant

bashbug Un script shell qui aide l'utilisateur à composer et à envoyer des courriers électroniques contenant des rapports de bogues 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-2.4.7

Le paquet Libtool contient le script de prise en charge générique des bibliothèques de GNU. Il cache la complexité d'utilisation des bibliothèques partagées dans une interface cohérente et portable.

Temps de construction approximatif: 1.5 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 significativement réduit 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 échouent dans l'environnement de construction de LFS à cause d'une dépendance circulaire, mais les tests passent tous si 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-1.23

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: 0.1 SBU
Espace disque requis: 13 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
```

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: 6.0 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 lancez :

```
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-2.4.8

Le paquet Expat contient une bibliothèque C orientée flux pour analyser du XML.

Temps de construction approximatif: 0.1 SBU
Espace disque requis: 12 Mo

8.38.1. Installation d'Expat

Préparez la compilation d'Expat :

```
./configure --prefix=/usr    \
            --disable-static \
            --docdir=/usr/share/doc/expat-2.4.8
```

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,css} /usr/share/doc/expat-2.4.8
```

8.38.2. Contenu d'Expat

Programme installé: xmlwf
Bibliothèques installées: libexpat.so
Répertoire installé: /usr/share/doc/expat-2.4.8

Descriptions courtes

xmlwf Est un outil de validation pour vérifier si les documents XML sont bien formés ou non

`libexpat` Contient les fonctions de l'API de l'analyse XML

8.39. Inetutils-2.2

Le paquet Inetutils contient des programmes réseaux basiques.

Temps de construction approximatif: 0.3 SBU
Espace disque requis: 30 Mo

8.39.1. Installation de Inetutils

Préparez la compilation d'Inetutils :

```
./configure --prefix=/usr        \
            --bindir=/usr/bin    \
            --localstatedir=/var \
            --disable-logger     \
            --disable-whois      \
            --disable-rcp        \
            --disable-rexec      \
            --disable-rlogin     \
            --disable-rsh        \
            --disable-servers
```

Voici la signification des options de configuration :

`--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 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
```

Installez le paquet :

```
make install
```

Déplacez un programme au bon emplacement :

```
mv -v /usr/{,s}bin/ifconfig
```

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 le 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. Less-590

Le paquet Less contient un visualiseur de fichiers texte.

Temps de construction approximatif: moins de 0.1 SBU
Espace disque requis: 4.2 Mo

8.40.1. Installation de Less

Préparez la compilation de Less :

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

Voici la signification des options de configuration :

`--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
```

8.40.2. Contenu de Less

Programmes installés: less, lessecho et lesskey

Descriptions courtes

less Un visualiseur de fichiers. Il affiche le contenu d'un fichier, vous permettant de le faire défiler, de chercher des chaînes et de sauter vers des repères

lessecho Nécessaire pour étendre les méta-caractères, comme * et ?, dans les noms de fichiers sur les systèmes Unix

lesskey Utilisé pour spécifier les associations de touches pour **less**

8.41. Perl-5.36.0

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

Temps de construction approximatif: 9.3 SBU
Espace disque requis: 226 Mo

8.41.1. Installation de Perl

Cette version de Perl compile à présent 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. Autrement, exécutez l'exacte commande ci-dessous pour utiliser les paramètres par défaut que détecte Perl automatiquement :

```
sh Configure -des                                         \
             -Dprefix=/usr                                \
             -Dvendorprefix=/usr                          \
             -Dprivlib=/usr/lib/perl5/5.36/core_perl      \
             -Darchlib=/usr/lib/perl5/5.36/core_perl      \
             -Dsitelib=/usr/lib/perl5/5.36/site_perl      \
             -Dsitearch=/usr/lib/perl5/5.36/site_perl     \
             -Dvendorlib=/usr/lib/perl5/5.36/vendor_perl  \
             -Dvendorarch=/usr/lib/perl5/5.36/vendor_perl \
             -Dman1dir=/usr/share/man/man1                \
             -Dman3dir=/usr/share/man/man3                \
             -Dpager="/usr/bin/less -isR"                 \
             -Duseshrplib                                 \
             -Dusethreads
```

Voici la signification des options de configuration :

`-Dvendorprefix=/usr`
Ceci s'assure que **perl** sait communiquer aux paquets l'emplacement 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`
Étant donné que Groff n'est pas encore installé, **Configure** pense que nous ne voulons pas des 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 Perl (5.36), qui permet de mettre Perl à jour vers de nouvelles versions Patch (5.36.0) sans avoir besoin de réinstaller tous les modules.

Compilez le paquet :

```
make
```

Pour tester les résultats (approximativement 11 SBU), lancez :

```
make test
```

Note

Nine tests fail due to using the most recent version of zlib.

Installez le paquet et faites le ménage :

```
make install
unset BUILD_ZLIB BUILD_BZIP2
```

8.41.2. Contenu de Perl

Programmes installés:	corelist, cpan, enc2xs, encguess, h2ph, h2xs, instmodsh, json_pp, libnetcfg, perl, perl5.36.0 (lien matériel vers perl), perlbug, perldoc, perlivp, perlthanks (lien matériel 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
perl5.36.0	Un lien matériel 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

8.42. 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

8.42.1. 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
```

8.42.2. Contenu de XML::Parser

Module installé: Expat.so

Descriptions courtes

`Expat` fournit l'interface Perl avec Expat

8.43. 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

8.43.1. 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
```

8.43.2. 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

8.44. 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 6.8 SBU avec les tests)
Espace disque requis: 24 Mo

8.44.1. Installation de Autoconf

Préparez la compilation d'Autoconf :

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

Compilez le paquet :

```
make
```

Pour tester les résultats, lancez :

```
make check
```

Note

Le temps de test d'autoconf 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.

Installez le paquet :

```
make install
```

8.44.2. 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 qui configurent 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 modèles 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 sont effectuées 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, tout en cherchant des problèmes de portabilité communs, 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 utiliser les noms de macros actuels

ifnames aide à écrire les fichiers `configure.in` pour un paquet logiciel. Il affiche les identifiants 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**

8.45. Automake-1.16.5

Le paquet Automake contient des programmes de génération de Makefile à utiliser avec Autoconf.

Temps de construction approximatif: moins de 0.1 SBU (environ 8.3 SBU avec les tests)
Espace disque requis: 115 Mo

8.45.1. Installation de Automake

Préparez la compilation d'Automake :

```
./configure --prefix=/usr --docdir=/usr/share/doc/automake-1.16.5
```

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
```

Le test t/subobj.sh est connu pour échouer.

Installez le paquet :

```
make install
```

8.45.2. Contenu de Automake

Programmes installés: aclocal, aclocal-1.16 (lié matériel avec aclocal), automake, et automake-1.16 (lié matériel avec automake)
Répertoires installés: /usr/share/aclocal-1.16, /usr/share/automake-1.16, et /usr/share/doc/automake-1.16.5

Descriptions courtes

aclocal Génère des fichiers `aclocal.m4` basés sur le contenu du fichier `configure.in`

aclocal-1.16 Un lien matériel 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 matériel vers **automake**

8.46. OpenSSL-3.0.3

Le paquet OpenSSL contient des outils de gestion et des bibliothèques cryptographiques. 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: 5.4 SBU
Espace disque requis: 474 Mo

8.46.1. Installation d'OpenSSL

Préparez la compilation d'OpenSSL :

```
./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 dépend de valeurs incohérentes des paramètres CONFIG_CRYPTO_USER_API*). S'il échoue, vous pouvez l'ignorer.

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-3.0.3
```

Si vous le souhaitez, installez de la documentation supplémentainre :

```
cp -vfr doc/* /usr/share/doc/openssl-3.0.3
```

Note

You should update OpenSSL when a new version which fixes vulnerabilities is announced. Since OpenSSL 3.0.0, the OpenSSL versioning scheme follows the MAJOR.MINOR.PATCH format. API/ABI compatibility are guaranteed for the same MAJOR version number. Because LFS installs only the shared libraries, there is no need to recompile packages which link to `libcrypto.so` or `libssl.so` *when upgrading to a version with MAJOR version number unchanged*.

Cependant, tout programme en cours d'exécution lié à ces bibliothèque doit être arrêté et redémarré. Lisez les sections en rapport à ce problème dans Section 8.2.1, « Problèmes de mise à jour » pour plus de détails.

8.46.2. 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-3.0.3

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**

8.47. Kmod-29

Le paquet Kmod contient des bibliothèques et des outils pour charger les modules du noyau

Temps de construction approximatif: 0.1 SBU
Espace disque requis: 12 Mo

8.47.1. Installation de Kmod

Préparez la compilation de Kmod :

```
./configure --prefix=/usr          \
            --sysconfdir=/etc      \
            --with-openssl         \
            --with-xz              \
            --with-zstd            \
            --with-zlib
```

Voici la signification des options de configuration :

`--with-openssl`
Cette option permet à Kmod de gérer les signatures PKCS7 des modules du noyau.

`--with-xz`, *`--with-zlib`* et *`--with-zstd`*
Ces options permettent à Kmod de gérer les modules noyau compressés.

Compilez le paquet :

```
make
```

La suite de tests de ce paquet nécessite les en-têtes brutes (pas les en-têtes du noyau « nettoyées » installés plus tôt), qui sont en-dehors des buts de LFS.

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 modinfo modprobe rmmod; do
  ln -sfv ../bin/kmod /usr/sbin/$target
done

ln -sfv kmod /usr/bin/lsmod
```

8.47.2. Contenu de Kmod

Programmes installés: depmod (lien vers kmod), insmod (lien vers kmod), kmod, lsmod (lien vers kmod), modinfo (lien vers kmod), modprobe (lien vers kmod) et rmmod (lien 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 les modules du noyau

lsmod Liste les modules actuellement chargés

modinfo Examine un fichier objet associé à un module du noyau et affiche toute information récoltée

modprobe Utilise un fichier de dépendance, créé par **depmod**, pour charger automatiquement les modules adéquats

rmmod Décharge les modules du noyau en cours d'exécution

`libkmod` Cette bibliothèque est utilisée par d’autres programmes pour charger et décharger les modules du noyau

8.48. Libelf de Elfutils-0.187

Libelf est une bibliothèque pour gérer les fichiers ELF (Executable and Linkable Format).

Temps de construction approximatif: 0.9 SBU
Espace disque requis: 116 Mo

8.48.1. Installation de Libelf

Libelf fait partie du paquet elfutils-0.187. Utilisez elfutils-0.187.tar.bz2 comme archive des sources.

Préparez la compilation de Libelf :

```
./configure --prefix=/usr                \
            --disable-debuginfod         \
            --enable-libdebuginfod=dummy
```

Compilez le paquet :

```
make
```

Pour tester les résultats, lancez :

```
make check
```

Installez uniquement Libelf :

```
make -C libelf install
install -vm644 config/libelf.pc /usr/lib/pkgconfig
rm /usr/lib/libelf.a
```

8.48.2. Contenu de Libelf

Bibliothèque installée: libelf.so (lien symbolique) et libelf-0.187.so
Répertoire installé: /usr/include/elfutils

Descriptions courtes

`libelf` Contient les fonction de l'API pour gérer les fichiers objet ELF

8.49. Libffi-3.4.2

La bibliothèque Libffi fournit une interface de programmation portable et de haut niveau pour diverses conventions d'appel. Cela permet au programmeur d'appeler des fonctions par la description de leur interface d'appel à l'exécution.

Temps de construction approximatif: 1.9 SBU
Espace disque requis: 10 Mo

8.49.1. Installation de Libffi

Note

Similar to GMP, libffi builds with optimizations specific to the processor in use. If building for another system, change the value of the *--with-gcc-arch=* parameter in the following command to an architecture name fully implemented by the CPU on that system. If this is not done, all applications that link to libffi will trigger Illegal Operation Errors.

Préparez la compilation de libffi :

```
./configure --prefix=/usr          \
            --disable-static       \
            --with-gcc-arch=native \
            --disable-exec-static-tramp
```

Voici la signification de l'option de configuration :

--with-gcc-arch=native
Vérifie 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 ce dernier dans le paramètre. Pour des détails concernant les types de systèmes alternatifs, voyez *les options x86 dans le manuel de GCC*.

--disable-exec-static-tramp
Disable static trampoline support. It's a new security feature in libffi, but some BLFS packages (notably GJS) have not been adapted for it.

Compilez le paquet :

```
make
```

Pour tester les résultats, lancez :

```
make check
```

Installez le paquet :

```
make install
```

8.49.2. Contenu de Libffi

Bibliothèque installée: libffi.so

Descriptions courtes

`libffi` contient les fonctions de l'API de l'interface pour les fonctions externes.

8.50. Python-3.10.5

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: 4.3 SBU
Espace disque requis: 275 Mo

8.50.1. Installation de Python 3

Préparez la compilation de Python :

```
./configure --prefix=/usr        \
            --enable-shared      \
            --with-system-expat  \
            --with-system-ffi    \
            --enable-optimizations
```

Voici la signification des options de configuration :

`--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.

`--enable-optimizations`
Ce paramètre active les optimisations stables, mais coûteuses.

Compilez le paquet :

```
make
```

Lancer les tests n'est pour l'instant pas recommandé. Les tests sont connus pour bloquer indéfiniment dans l'environnement partiel de LFS. Si vous le souhaitez, vous pouvez relancer les tests à la fin de ce chapitre ou quand vous réinstallerez Python 3 dans BLFS. Pour tout de même lancer les tests, lancez **make test**.

Installez le paquet :

```
make install
```

In serveral places we use the **pip3** command to install Python 3 programs and modules for all users as `root`. This conflicts with the Python developers recommendation to install packages into a virtual environment or the home directory of a regular user (by running **pip3** as this user). To this end, a multi-line warning is written when using **pip3** as the `root` user. The main reason of this recommendation is for avoiding a conflict with the system package manager (**dpkg** for example), but LFS does not have a system-wide package manager so this is not a problem. If desired, suppress this warning by running the following command:

```
sed -e '/def warn_if_run_as_root/a\    return' \
    -i /usr/lib/python3.10/site-packages/pip/_internal/cli/req_command.py
```

Important

In LFS and BLFS we normally build and install Python modules with the **pip3** command. Please take care that the **pip3 install** commands in both the books should be run as the `root` user unless it's for a Python virtual environment. Running a **pip3 install** as a non-`root` user may seem to work fine, but it will cause the installed module to be inaccessible by other users.

pip3 install will not reinstall an already installed module by default. For using the **pip3 install** command to upgrade a module (for example, from meson-0.61.3 to meson-0.62.0), insert the option *`--upgrade`* into the command line. If it's really necessary to downgrade a module or reinstall the same version for some reason, insert *`--force-reinstall`* into the command line.

Si vous le souhaitez, installez la documentation préformatée :

```
install -v -dm755 /usr/share/doc/python-3.10.5/html

tar --strip-components=1  \
    --no-same-owner       \
    --no-same-permissions \
    -C /usr/share/doc/python-3.10.5/html \
    -xvf ../python-3.10.5-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.

8.50.2. Contenu de Python 3

Programmes installés: 2to3, idle3, pip3, pydoc3, python3 et python3-config
Bibliothèques installées: libpython3.10.so et libpython3.so
Répertoires installés: /usr/include/python3.10, /usr/lib/python3 et /usr/share/doc/python-3.10.5

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

8.51. Wheel-0.37.1

Wheel is a Python library that is the reference implementation of the Python wheel packaging standard.

Temps de construction approximatif: moins de 0.1 SBU
Espace disque requis: 268 Ko

8.51.1. Installation of Wheel

Install wheel with the following command:

```
pip3 install --no-index $PWD
```

The meaning of the pip3 options:

install
Install the package.

`--no-index`
Prevent pip from fetching files from the online package repository (PyPI). If packages are installed in the correct order, then it won't need to fetch any files in the first place, but this option adds some safety in case of user error.

`$PWD`
Look for files to install in the current working directory.

8.51.2. Contents of Wheel

Installed program: wheel
Installed directory: /usr/lib/python3.10/site-packages/wheel

Short Descriptions

wheel is an utility to unpack, pack, or convert wheel packages

8.52. Ninja-1.11.0

Ninja est un petit système de construction qui met l'accent sur la rapidité.

Temps de construction approximatif: 0.2 SBU
Espace disque requis: 64 Mo

Astuce

Cette section n'est pas obligatoirement requise pour LFS si vous n'utilisez pas systemd. Cependant, ninja associé à meson devient un système de construction puissant, utilisé de plus en plus fréquemment. Il est requis par plusieurs paquets dans *le livre BLFS*.

8.52.1. Installation de Ninja

Lorsqu'il est exécuté, 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 exécuté 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/ninja
install -vDm644 misc/zsh-completion  /usr/share/zsh/site-functions/_ninja
```

8.52.2. Contenu de Ninja

Programmes installés: ninja

Descriptions courtes

ninja est le système de construction Ninja

8.53. Meson-0.62.2

Meson est un système de construction libre conçu pour être très rapide et aussi facile à utiliser que possible.

Temps de construction approximatif: moins de 0.1 SBU
Espace disque requis: 41 Mo

Astuce

Cette section n'est pas strictement requise pour LFS si vous n'utilisez pas systemd. Cependant, meson/ninja est un système de construction puissant qui sera utilisé de plus en plus souvent. Il est requis pour plusieurs paquets dans *le livre BLFS*.

8.53.1. Installation de Meson

Compilez Meson avec la commande suivante :

```
pip3 wheel -w dist --no-build-isolation --no-deps $PWD
```

The test suite requires some packages out of the scope of LFS.

Installez le paquet :

```
pip3 install --no-index --find-links dist meson
install -vDm644 data/shell-completions/bash/meson /usr/share/bash-completion/completions/meson
install -vDm644 data/shell-completions/zsh/_meson /usr/share/zsh/site-functions/_meson
```

Voici la signification des paramètres d'installation :

`-w dist`
Put the created wheels into the `dist` directory.

`--find-links dist`
Install wheels from the `dist` directory.

8.53.2. Contenu de Meson

Programmes installés: meson
Répertoire installé: /usr/lib/python3.10/site-packages/meson-0.62.2.dist-info and /usr/lib/python3.10/site-packages/mesonbuild

Descriptions courtes

meson Un système de construction pour une plus grande productivité

8.54. Coreutils-9.1

Le paquet Coreutils contient des outils d'affichage de de configuration des caractéristiques de base d'un système.

Temps de construction approximatif: 2.6 SBU
Espace disque requis: 153 Mo

8.54.1. Installation de Coreutils

POSIX exige que les programmes de Coreutils reconnaissent correctement les limites de caractères même pour des encodages sur plusieurs octets. Le correctif suivant corrige cette absence de conformité et d'autres bogues liés à l'internationalisation.

```
patch -Np1 -i ../coreutils-9.1-i18n-1.patch
```

Note

Autrefois, on a pu trouver beaucoup de bogues dans ce correctif. Lorsque vous signalez aux mainteneurs de Coreutils de nouveaux bogues, merci de vérifier d'abord qu'ils peuvent être reproduits sans ce correctif.

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 la partie « 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
```

Fix some of the permissions so that the non-`root` user can compile and run the tests:

```
chown -Rv tester .
```

Maintenant lancez les tests :

```
su tester -c "PATH=$PATH make RUN_EXPENSIVE_TESTS=yes check"
```

The sort-NaN-infloop test is known to fail with GCC-12.

Supprimez le groupe temporaire :

```
sed -i '/dummy/d' /etc/group
```

Installez le paquet :

```
make install
```

Déplacez certains programmes aux emplacements spécifiés par le FHS :

```
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
```

8.54.2. 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 d'un nom de fichier

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 en tant que 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 et affiche sur trois colonnes les lignes uniques et les lignes communes

cp Copie des fichiers

csplit Divise un fichier donné en plusieurs fichiers indiqués et les sépare par des motifs donnés ou des numéros de lignes. Il affiche le nombre total d'octets pour chaque nouveau fichier

cut Affiche des parties de lignes et sélectionne 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 dont 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, comme prévu, et 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 donné

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 matériel avec le nom donné vers le fichier indiqué

ln Crée des liens symboliques ou matériels entre des fichiers

logname Indique l'identifiant 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 de périphérique est de type caractère, bloc, ou 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 qui affiche certaines 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 les répertoires 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 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 éléments, 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 et les 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, comme prévu, et quitte toujours avec un code de sortie indiquant une réussite
truncate	Réduit ou augmente un fichier selon la taille spécifiée
tsort	Réalise un tri topologique et 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**

8.55. Check-0.15.2

Check est un environnement de tests unitaires pour C.

Temps de construction approximatif: 0.1 SBU (environ 3.8 SBU avec les tests)
Espace disque requis: 12 Mo

8.55.1. Installation de Check

Préparez la compilation de Check :

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

Construisez le paquet :

```
make
```

La compilation est maintenant terminée. Pour lancer la suite de tests de Check, lancez la commande suivante :

```
make check
```

Installez le paquet :

```
make docdir=/usr/share/doc/check-0.15.2 install
```

8.55.2. Contenu de Check

Programme installé: checkmk
Bibliothèque installÃé: libcheck.so

Descriptions courtes

checkmk Script awk pour générer des tests unitaires C à utiliser avec l'environnement de tests unitaires de Check

`libcheck.so` Contient les fonctions permettant à Check d'être appelé depuis un programme de test

8.56. Diffutils-3.8

Le paquet Diffutils contient des programmes qui affichent les différences entre fichiers ou répertoires.

Temps de construction approximatif: 0.6 SBU
Espace disque requis: 34 Mo

8.56.1. 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
```

8.56.2. 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

8.57. Gawk-5.1.1

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

Temps de construction approximatif: 0.4 SBU
Espace disque requis: 43 Mo

8.57.1. 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 vous le souhaitez, installez la documentation :

```
mkdir -pv                                   /usr/share/doc/gawk-5.1.1
cp    -v doc/{awkforai.txt,*.{eps,pdf,jpg}} /usr/share/doc/gawk-5.1.1
```

8.57.2. Contenu de Gawk

Programmes installés: awk (lien vers gawk), gawk et awk-5.1.1
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-5.1.1

Descriptions courtes

awk Un lien vers **gawk**

gawk Un programme de manipulation de fichiers texte. C'est l'implémentation GNU d'**awk**

gawk-5.1.1 Un lien matériel vers **gawk**

8.58. Findutils-4.9.0

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: 51 Mo

8.58.1. Installation de Findutils

Préparez la compilation de Findutils :

```
case $(uname -m) in
    i?86)   TIME_T_32_BIT_OK=yes ./configure --prefix=/usr --localstatedir=/var/lib/locate ;;
    x86_64) ./configure --prefix=/usr --localstatedir=/var/lib/locate ;;
esac
```

Voici la signification des options de configuration :

TIME_32_BIT_OK=yes
Ce paramètre est requis pour la construction sur un système 32 bits.

`--localstatedir`
Cette option modifie l'emplacement de la base de données **locate** pour qu'elle soit dans `/var/lib/locate`, ce qui est compatible avec le 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
```

8.58.2. 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 ceux qui contiennent une certaine chaîne ou qui correspondent à un certain modèle

updatedb Met à jour la base de données **locate**. Il parcourt le système de fichiers entier (y compris les autres systèmes de fichiers actuellement montés, sauf si le contraire est spécifié) et place tous les noms de fichiers qu'il trouve dans la base de données

xargs Peut être utilisé pour lancer une commande donnée sur une liste de fichiers

8.59. Groff-1.22.4

Le paquet Groff contient des programmes de formatage de texte.

Temps de construction approximatif: 0.5 SBU
Espace disque requis: 88 Mo

8.59.1. 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. Même 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.

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
```

8.59.2. 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 recadrée

gdiffmk Marque les différences entre des fichiers groff/nroff/troff

glilypond Transforme les partitions de musiques du langage lilypond en langage groff

gperl Préprocesseur 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 un 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 Est un pilote **groff** pour les imprimantes Canon CAPSL (imprimantes laser de la série LBP-4 et LBP-8)

grolj4 Est un pilote pour **groff** produisant une sortie au format PCL5 adapté aux 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, utilisé pour 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 répète ce processus jusqu'à la fin de l'entrée

mmroff Un simple 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 Est une 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` vers le format ASCII

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 recadrée

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 en fichiers DVI
roff2html	Transforme les fichiers roff en fichiers HTML
roff2pdf	Transforme les fichiers roff en fichiers PDF
roff2ps	Transforme les fichiers roff en fichiers ps
roff2text	Transforme les fichiers roff en fichiers textes
roff2x	Transforme les fichiers roff vers 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

8.60. GRUB-2.06

Le paquet Grub contient un chargeur de démarrage, le *GRand Unified Bootloader*.

Temps de construction approximatif: 0.7 SBU
Espace disque requis: 158 Mo

8.60.1. Installation de GRUB

Note

Si votre système prend en charge l'UEFI et que vous souhaitez démarrer LFS avec l'UEFI, vous pouvez sauter ce paquet dans LFS et installer GRUB avec la prise en charge de l'UEFI (et ses dépendances) en suivant *la page BLFS* à la fin de ce chapitre.

Préparez la compilation de GRUB :

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

Les suites de tests de ces paquets ne sont pas recommandées. La plupart des tests dépendent de paquets qui ne sont pas disponibles dans l'environnement LFS limité. Pour lancer les tests malgré tout, lancez **make check**.

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 ».

8.60.2. 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

8.61. Gzip-1.12

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

Temps de construction approximatif: 0.1 SBU
Espace disque requis: 20 Mo

8.61.1. 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
```

8.61.2. Contenu de Gzip

Programmes installés: gunzip, gzexe, gzip, uncompress (lien matériel vers 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`

8.62. IPRoute2-5.18.0

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

8.62.1. 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-svn/server/db.html*.

```
sed -i /ARPD/d Makefile
rm -fv man/man8/arpd.8
```

Compilez le paquet :

```
make NETNS_RUN_DIR=/run/netns
```

Ce paquet n'a pas de suite de tests fonctionnelle.

Installez le paquet :

```
make SBINDIR=/usr/sbin install
```

Si vous le souhaitez, installez la documentation :

```
mkdir -pv             /usr/share/doc/iproute2-5.18.0
cp -v COPYING README* /usr/share/doc/iproute2-5.18.0
```

8.62.2. 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/iproute2-5.18.0

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

8.63. Kbd-2.5.0

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

8.63.1. Installation de Kbd

Le comportement des touches Retour Arrière 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-2.5.0-backspace-1.patch
```

Après la correction, la touche Retour Arrière 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 de l'option 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 vous le souhaitez, installez la documentation :

```
mkdir -pv           /usr/share/doc/kbd-2.5.0
cp -R -v docs/doc/* /usr/share/doc/kbd-2.5.0
```

8.63.2. 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-2.5.0 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 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 Définit 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

8.64. Libpipeline-1.5.6

Le paquet Libpipeline contient une bibliothèque pour manipuler des pipelines (tubes) de sous-processus de façon flexible et commode.

Temps de construction approximatif: 0.1 SBU
Espace disque requis: 9.7 Mo

8.64.1. 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
```

8.64.2. Contenu de Libpipeline

Bibliothèque installée: libpipeline.so

Descriptions courtes

`libpipeline` Cette bibliothèque est utilisée pour construire de façon sécurisée des pipelines entre des sous-processus

8.65. 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: 0.5 SBU
Espace disque requis: 13 Mo

8.65.1. 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
```

8.65.2. 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

8.66. 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 ») généralement créé par le programme **diff**.

Temps de construction approximatif: 0.2 SBU
Espace disque requis: 12 Mo

8.66.1. 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
```

8.66.2. 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.

8.67. 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: 1.7 SBU
Espace disque requis: 40 Mo

8.67.1. Installation de Tar

Préparez la compilation de Tar :

```
FORCE_UNSAFE_CONFIGURE=1  \
./configure --prefix=/usr
```

Voici la signification de l'option de configuration :

`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, lancez :

```
make check
```

Un test, capabiblities: binary store/restore, est connu pour échouer s'il est lancé (LFS n'a pas selinux), mais sera passé si le noyau hôte ne prend pas en charge les attributs étendus sur le système de fichiers utilisé pour la construction de LFS.

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-6.8

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

Temps de construction approximatif: 0.6 SBU
Espace disque requis: 112 Mo

8.68.1. Installation de Texinfo

Préparez la compilation de Texinfo :

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

De nouveau, corrigez un problème de construction du paquet avec Glibc-2.34 et supérieur :

```
sed -e 's/__attribute_nonnull__/__nonnull/' \
    -i gnulib/lib/malloc/dynarray-skeleton.c
```

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-8.2.5014

Le paquet Vim contient un puissant éditeur de texte.

Temps de construction approximatif: 2.4 SBU
Espace disque requis: 206 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-svn/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-8.2.5014`, en cohérence avec l'emplacement de la documentation d'autres paquets :

```
ln -sv ../vim/vim82/doc /usr/share/doc/vim-8.2.5014
```

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-svn/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"
" Début de /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

" Fin de /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. Eudev-3.2.11

Le paquet Eudev contient des programmes pour la création dynamique de nœuds de périphériques.

Temps de construction approximatif: 0.2 SBU
Espace disque requis: 83 Mo

8.70.1. Installation d'Eudev

Préparez la compilation d'Eudev :

```
./configure --prefix=/usr           \
            --bindir=/usr/sbin      \
            --sysconfdir=/etc       \
            --enable-manpages       \
            --disable-static
```

Compilez le paquet :

```
make
```

Créez des répertoires nécessaires pour les tests, mais qui feront aussi partie de l'installation :

```
mkdir -pv /usr/lib/udev/rules.d
mkdir -pv /etc/udev/rules.d
```

Pour tester les résultats, lancez :

```
make check
```

Installez le paquet :

```
make install
```

Installez quelques règles personnalisées et des fichiers de prise en charge utiles dans un environnement LFS :

```
tar -xvf ../udev-lfs-20171102.tar.xz
make -f udev-lfs-20171102/Makefile.lfs install
```

8.70.2. Configuration d'Eudev

Les informations sur les périphériques sont stockées dans les répertoires `/etc/udev/hwdb.d` et `/usr/lib/udev/hwdb.d`. Eudev a besoin de compiler ces informations dans une base de données binaire `/etc/udev/hwdb.bin`. Créez la base de données initiale :

```
udevadm hwdb --update
```

Vous devez lancer cette commande à chaque mise à jour des informations sur le matériel.

8.70.3. Contenu d'Eudev

Programmes installés: udevadm et udevd
Bibliothèques installées: libudev.so
Répertoires installés: /etc/udev, /usr/lib/udev et /usr/share/doc/udev-udev-lfs-20171102

Descriptions courtes

udevadm Outil d'administration générique d'udev : il contrôle le démon udevd, fournit des informations à partir d'une base de données Udev, surveille les uevents, attend la fin de certains uevents, teste la configuration d'Udev et déclenche des uevents pour un périphérique donné

udevd Un démon qui écoute les uevents sur le socket netlink, crée des périphériques et lance les programmes externes configurés en réponse à ces uevents

`libudev` Une interface bibliothèque avec les informations de périphérique d'udev

`/etc/udev` Contient les fichiers de configuration d'Udev, les autorisations des périphériques et les règles de périphériques, et les règles pour le nommage des périphériques

8.71. Man-DB-2.10.2

Le paquet Man-DB contient des programmes pour chercher et voir des pages de manuel.

Temps de construction approximatif: 0.3 SBU
Espace disque requis: 39 Mo

8.71.1. Installation de Man-DB

Préparez la compilation de man-DB :

```
./configure --prefix=/usr                         \
            --docdir=/usr/share/doc/man-db-2.10.2 \
            --sysconfdir=/etc                     \
            --disable-setuid                      \
            --enable-cache-owner=bin              \
            --with-browser=/usr/bin/lynx          \
            --with-vgrind=/usr/bin/vgrind         \
            --with-grap=/usr/bin/grap             \
            --with-systemdtmpfilesdir=            \
            --with-systemdsystemunitdir=
```

Voici la signification des options de configuration :

`--disable-setuid`
Ceci empêche que le programme **man** se voit attribué l'ID de l'utilisateur `man`.

`--enable-cache-owner=bin`
Cela attribue 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 pouvoir les installer vous-même après avoir fini LFS si vous le souhaitez.

`--with-systemd...`
Ces paramètres évitent d'installer des répertoires et des fichiers de systemd inutiles.

Compilez le paquet :

```
make
```

Pour tester les résultats, lancez :

```
make check
```

Installez le paquet :

```
make install
```

8.71.2. Pages de manuel non anglophones 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 sont 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	Grec (el)	ISO-8859-7
Islandais (is)	ISO-8859-1	Slovaque (sk)	ISO-8859-2
Italien (it)	ISO-8859-1	Slovène (sl)	ISO-8859-2
Norvégien bokmål (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 simplifié, Singapour (zh_SG)	GBK
Bulgare (bg)	CP1251	Chinois traditionnel, Hong Kong (zh_HK)	BIG5HKSCS
Tchèque (cs)	ISO-8859-2	Chinois traditionnel (zh_TW)	BIG5

Note

Les pages de manuel dans des langues non incluses dans la liste ne sont pas prises en charge.

8.71.3. Contenu de Man-DB

Programmes installés: accessdb, apropos (lien vers whatis), catman, lexgrog, man, man-recode, 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-2.10.2

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 sous forme de résumé d'une ligne à propos d'une page de manuel donnée
man	Formate et affiche les pages de manuel demandées
man-recode	Converti les pages de manuel vers un autre encodage
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 man.conf et 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 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**

8.72. Procps-4.0.0

Le paquet Procps-ng contient des programmes pour surveiller les processus.

Temps de construction approximatif: 0.4 SBU
Espace disque requis: 19 Mo

8.72.1. Installation de Procps-ng

Préparez maintenant la compilation de procps-ng :

```
./configure --prefix=/usr                            \
            --docdir=/usr/share/doc/procps-ng-4.0.0 \
            --disable-static                         \
            --disable-kill
```

Voici la signification de l'option de configuration :

`--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
```

One test named `free with commit` may fail if some applications with a custom memory allocator (for example, JVM and Web browsers) are running on the host distro.

Installez le paquet :

```
make install
```

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: libproc-2.so
Répertoires installés: /usr/include/procps and /usr/share/doc/procps-ng-4.0.0

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

`libproc-2` Contient les fonctions utilisées par la plupart des programmes de ce paquet

8.73. Util-linux-2.38

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: 261 Mo

8.73.1. Installation d'Util-linux

Préparez la compilation d'Util-linux :

```
./configure ADJTIME_PATH=/var/lib/hwclock/adjtime   \
            --bindir=/usr/bin    \
            --libdir=/usr/lib    \
            --sbindir=/usr/sbin  \
            --docdir=/usr/share/doc/util-linux-2.38 \
            --disable-chfn-chsh  \
            --disable-login      \
            --disable-nologin    \
            --disable-su         \
            --disable-setpriv    \
            --disable-runuser    \
            --disable-pylibmount \
            --disable-static     \
            --without-python     \
            --without-systemd    \
            --without-systemdsystemunitdir
```

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
```

If desired, run the test suite as a non-`root` user:

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"
```

The *hardlink* tests will fail if the host's kernel does not have the option CONFIG_CRYPTO_USER_API_HASH set.

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, fdisk, fincore, findfs, findmnt, flock, fsck, fsck.cramfs, fsck.minix, fsfreeze, fstrim, getopt, hexdump, hwclock, i386, ionice, ipcmk, ipcrm, ipcs, irqtop, isosize, kill, last, lastb (lien vers last), ldattach, linux32, linux64, logger, look, losetup, lsblk, lscpu, lsipc, lsirq, lslocks, lslogins, lsmem, lsns, mcookie, mesg, mkfs, mkfs.bfs, mkfs.cramfs, mkfs.minix, mkswap, more, mount, mountpoint, namei, nsenter, partx, pivot_root, prlimit, readprofile, rename, renice, resizepart, rev, rfkill, rtcwake, script, scriptlive, scriptreplay, setarch, setsid, setterm, sfdisk, sulogin, swaplabel, swapoff (lien vers swapon), swapon, switch_root, taskset, uclampset, 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-2.38 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

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

irqtop Affiche le compteur d'interruption noyau dans un affichage similaire à `top(1)`

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

lsirq Affiche les information du compteur d'interruption du noyau

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
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
scriptlive	Rejoue des scripts type de session en utilisant les informations de temps
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

taskset Récupère ou initialise un processus vis-à-vis du processeur

uclampset Manipule les attributs de verrouillage d'utilisation du système ou d'un processus

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-1.46.5

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, 1.3 SBU on an SSD
Espace disque requis: 93 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           \
             --sysconfdir=/etc       \
             --enable-elf-shlibs     \
             --disable-libblkid      \
             --disable-libuuid       \
             --disable-uuidd         \
             --disable-fsck
```

Voici la signification des options de configuration :

`--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, u_direct_io, est connu pour échouer sur certains systèmes.

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 permet également de changer les systèmes de fichiers `ext3`, la version journalisée des systèmes de fichiers `ext2`

compile_et Un compilateur de table d'erreurs qui 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 qui 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 Rend compte des 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 sauvegarder 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 à quel point un fichier particulier peut être mal fragmenté

fsck.ext2 Vérifie par défaut les systèmes de fichiers `ext2`. C'est un lien matériel vers **e2fsck**

fsck.ext3 Vérifie par défaut les systèmes de fichiers `ext3`. C'est un lien matériel vers **e2fsck**

fsck.ext4 Vérifie par défaut les systèmes de fichiers `ext4`. C'est un lien matériel vers **e2fsck**

logsave Sauvegarde la sortie d'une commande dans un fichier journal

lsattr Liste les attributs de fichiers sur un système de fichiers ext2

mk_cmds Convertit une table de noms de commandes et de messages d'aide en un fichier source C utilisable 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 matériel vers **mke2fs**

mkfs.ext3 Crée par défaut un système de fichiers `ext3`. C'est un lien matériel vers **mke2fs**

mkfs.ext4	Crée par défaut un système de fichiers `ext4`. C'est un lien matériel 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 courantes
`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. Sysklogd-1.5.1

Le paquet Sysklogd contient des programmes pour enregistrer les messages système, comme ceux donnés par le noyau lorsque des événements inhabituels surviennent.

Temps de construction approximatif: moins de 0.1 SBU
Espace disque requis: 0.6 Mo

8.75.1. Installation de Sysklogd

Tout d'abord, corrigez un problème qui crée une erreur de segmentation dans certaines conditions dans klogd et corrigez une conception obsolète du programme :

```
sed -i '/Error loading kernel symbols/{n;n;d}' ksym_mod.c
sed -i 's/union wait/int/' syslogd.c
```

Compilez le paquet :

```
make
```

Ce paquet n'est pas fourni avec une suite de tests.

Installez le paquet :

```
make BINDIR=/sbin install
```

8.75.2. Configuration de Sysklogd

Créez un nouveau fichier `/etc/syslog.conf` en lançant ce qui suit :

```
cat > /etc/syslog.conf << "EOF"
# Begin /etc/syslog.conf

auth,authpriv.* -/var/log/auth.log
*.*;auth,authpriv.none -/var/log/sys.log
daemon.* -/var/log/daemon.log
kern.* -/var/log/kern.log
mail.* -/var/log/mail.log
user.* -/var/log/user.log
*.emerg *

# End /etc/syslog.conf
EOF
```

8.75.3. Contenu de Sysklogd

Programmes installés: klogd et syslogd

Descriptions courtes

klogd Un démon système pour intercepter et tracer les messages du noyau

syslogd Enregistre les messages que les programmes systèmes donnent [Chaque message enregistré contient au moins une date et un nom d'hôte, et normalement aussi le nom du programme, mais cela dépend de la façon dont le démon de traçage effectue sa surveillance].

8.76. Sysvinit-3.04

Le paquet Sysvinit contient des programmes de contrôle du démarrage, de l'exécution et de l'arrêt de votre système.

Temps de construction approximatif: moins de 0.1 SBU
Espace disque requis: 1.4 Mo

8.76.1. Installation de Sysvinit

Tout d'abord, appliquez un correctif qui supprime plusieurs programmes installés par d'autres paquets, qui clarifie un message et qui corrige un avertissement du compilateur :

```
patch -Np1 -i ../sysvinit-3.04-consolidated-1.patch
```

Compilez le paquet :

```
make
```

Ce paquet n'est pas fourni avec une suite de tests.

Installez le paquet :

```
make install
```

8.76.2. Contenu de Sysvinit

Programmes installés: bootlogd, fstab-decode, halt, init, killall5, poweroff (liien vers halt), reboot (lien vers halt), runlevel, shutdown et telinit (link to init)

Descriptions courtes

bootlogd Trace les messages de démarrage dans le journal

fstab-decode Lance une commande avec les arguments de fstab-encoded (encodés à la fstab)

halt Lance normalement **shutdown** avec l'option *-h*, sauf s'il est déjà au niveau d'exécution 0, puis il demande au noyau d'arrêter le système. Mais, tout d'abord, il note dans le fichier `/var/log/wtmp` que le système est en cours d'arrêt

init Le premier processus à être exécuté lorsque le noyau a initialisé le matériel et qui prend la main sur le processus de démarrage et démarre tous les processus qui lui ont été indiqués dans son fichier de configuration

killall5 Envoie un signal à tous les processus sauf les processus de sa propre session, de façon à ne pas tuer le terminal parent

poweroff Indique au noyau d'arrêter le système et de couper l'ordinateur (voir **halt**)

reboot Indique au noyau de redémarrer le système (voir **halt**)

runlevel Indique le niveau d'exécution actuel et précédent comme précisé dans l'enregistrement du dernier niveau d'exécution dans `/run/utmp`

shutdown Arrête proprement le système en le signalant à tous les processus et à tous les utilisateurs connectés

telinit Indique à **init** dans quel niveau d'exécution entrer

8.77. 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.

8.78. Nettoyage

Cette section est facultative. Si l'utilisateur du système ne programme pas et ne prévoie pas de déboguer les logiciels du système, vous pouvez réduire la taille du système d'environ 2 Go en supprimant les symboles de débogage des binaires et des bibliothèques. Cela ne pose aucun problème en dehors du fait de ne plus pouvoir déboguer les logiciels complètement.

La plupart des gens utilisent les commandes mentionnées ci-dessous sans difficulté. Cependant, il est facile de faire une coquille et de rendre le nouveau système inutilisable, donc avant de lancer les commandes **strip**, il vaut mieux faire une sauvegarde du système LFS dans son état actuel.

Les symboles de débogage pour les bibliothèques choisies sont placés dans des fichiers séparés. Ces informations de débogage sont requises si vous lancez des tests de régression qui utilisent *valgrind* ou *gdb* plus tard dans BLFS.

Remarquez que **strip** remplacera le binaire ou la bibliothèque qu'il traite. Cela peut faire crasher les processus qui utilisent du code ou des données de ce fichier. Si le processus lançant **strip** lui-même est affecté, le binaire ou la bibliothèque en cours de nettoyage peut être détruit. Cela peut rendre le système complètement inutilisable. Pour éviter cela, nous copierons certains bibliothèques et binaires dans `/tmp`, les nettoierons là, et les installerons de nouveau avec la commande **install**. Lisez les entrées liées dans le Section 8.2.1, « Problèmes de mise à jour » pour les raisons d'utiliser la commande **install** ici.

Note

The ELF loader's name is ld-linux-x86-64.so.2 on 64-bit systems and ld-linux.so.2 on 32-bit systems. The construct below selects the correct name for the current architecture, excluding anything ending with « g », in case the commands below have already been run.

```
save_usrlib="$(cd /usr/lib; ls ld-linux*[^g])
             libc.so.6
             libthread_db.so.1
             libquadmath.so.0.0.0
             libstdc++.so.6.0.30
             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
    cp $LIB /tmp/$LIB
    strip --strip-unneeded /tmp/$LIB
    objcopy --add-gnu-debuglink=$LIB.dbg /tmp/$LIB
    install -vm755 /tmp/$LIB /usr/lib
    rm /tmp/$LIB
done

online_usrbin="bash find strip"
online_usrlib="libbfd-2.38.so
               libhistory.so.8.1
               libncursesw.so.6.3
               libm.so.6
               libreadline.so.8.1
               libz.so.1.2.12
               $(cd /usr/lib; find libnss*.so* -type f)"

for BIN in $online_usrbin; do
    cp /usr/bin/$BIN /tmp/$BIN
    strip --strip-unneeded /tmp/$BIN
    install -vm755 /tmp/$BIN /usr/bin
    rm /tmp/$BIN
done

for LIB in $online_usrlib; do
    cp /usr/lib/$LIB /tmp/$LIB
    strip --strip-unneeded /tmp/$LIB
    install -vm755 /tmp/$LIB /usr/lib
    rm /tmp/$LIB
done

for i in $(find /usr/lib -type f -name \*.so* ! -name \*dbg) \
         $(find /usr/lib -type f -name \*.a)                 \
         $(find /usr/{bin,sbin,libexec} -type f); do
    case "$online_usrbin $online_usrlib $save_usrlib" in
        *$(basename $i)* )
            ;;
        * ) strip --strip-unneeded $i
            ;;
    esac
done

unset BIN LIB save_usrlib online_usrbin online_usrlib
```

La commande rapportera un grand nombre de fichiers comme ayant une extension non reconnue. Vous pouvez ignorer ces avertissements sans problème. Cela signifie que ces fichiers sont des scripts et non des binaires.

8.79. Nettoyage

Enfin, nettoyez quelques fichiers laissés par l'exécution des tests :

```
rm -rf /tmp/*
```

Il y a aussi plusieurs fichiers installés dans les répertoires /usr/lib et /usr/libexec avec l'extension .la. Ce sont des fichiers « d'archive libtool ». Comme nous l'avons déjà dit, ils ne sont utiles que pour se lier statiquement à des bibliothèques. Ils ne sont pas requis, et éventuellement dangereux, lorsque vous utilisez des bibliothèque dynamiques partagées, surtout avec les systèmes de construction autres qu'autotools. Pour les supprimer, exécutez :

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

Pour plus d'information sur les fichiers d'archive libtool, voir la *section BLFS « À propos des fichiers d'archive libtool (.la) »*.

Le compilateur construit au Chapitre 6 et au Chapitre 7 est toujours partiellement installé mais n'est plus utile. Supprimez-le avec :

```
find /usr -depth -name $(uname -m)-lfs-linux-gnu\* | xargs rm -rf
```

Enfin, supprimez le compte utilisateur « tester » créé au début du chapitre précédent.

```
userdel -r tester
```

Chapitre 9. Configuration du système

9.1. Introduction

Le démarrage d'un système Linux englobe plusieurs tâches. Le processus implique de monter les systèmes de fichiers virtuels et réels, d'initialiser les périphériques, activer l'échange (swap), vérifier l'intégrité des systèmes de fichiers, monter les partitions ou les fichiers d'échange, régler l'horloge du système, activer le réseau, démarrer les démons nécessaires au système et accomplir d'autres tâches personnalisées dont a besoin l'utilisateur. Il faut organiser le processus pour s'assurer que les tâches s'effectuent dans l'ordre, mais en même temps, le plus vite possible.

9.1.1. System V

System V est le système de démarrage classique qu'on utilise dans les systèmes Unix et Unix-like comme Linux depuis 1983. Il consiste en un petit programme, **init**, qui initialise les programmes de base tels que **login** (via getty) et lance un script. Ce script, appelé en général **rc**, contrôle l'exécution d'un ensemble d'autres scripts effectuant les tâches nécessaires pour initialiser le système.

The **init** program is controlled by the `/etc/inittab` file and is organized into run levels that can be run by the user. In LFS, they are used as follows:

```
0 — halt
1 — Single user mode
2 — User definable
3 — Full multiuser mode
4 — User definable
5 — Full multiuser mode with display manager
6 — reboot
```

Le niveau d'exécution par défaut est en général 3 ou 5.

Avantages

- Système stabilisé et maîtrisé.
- Facile à personnaliser.

Inconvénients

- Peut être plus lent au démarrage. La vitesse moyenne d'un système LFS de base est de 8-12 secondes, le temps de démarrage étant mesuré à partir du premier message du noyau jusqu'à l'invite de connexion. La connectivité réseau arrive en général 2 secondes après l'invite de connexion.
- Gestion en série des tâches de démarrage. Ceci est lié au point précédent. La durée d'un processus comme la vérification d'un système de fichiers retardera tout le processus de démarrage.
- Ne prend pas directement en charge les fonctionnalités avancées comme les groupes de contrôle (cgroups) et l’ordonnancement à part équitable entre utilisateurs.
- L'ajout de scripts nécessite de prendre des décisions sur la séquence statique d'exécution à la main.

9.2. LFS-Bootscripts-20220327

Le paquet LFS-Bootscripts contient un ensemble de scripts pour démarrer ou arrêter le système LFS lors de l'amorçage ou de l'arrêt. Les fichiers de configuration et les procédures nécessaires à la personnalisation du processus de démarrage sont décrits dans les sections suivantes.

Temps de construction approximatif: moins de 0.1 SBU
Espace disque requis: BOOTSCRIPTS-INSTALL-Ko Ko

9.2.1. Installation de LFS-Bootscripts

Installez le paquet :

```
make install
```

9.2.2. Contenu de LFS-Bootscripts

Scripts installés: checkfs, cleanfs, console, functions, halt, ifdown, ifup, localnet, modules, mountfs, mountvirtfs, network, rc, reboot, sendsignals, setclock, ipv4-static, swap, sysctl, sysklogd, template, udev et udev_retry
Répertoires installés: /etc/rc.d, /etc/init.d (lien symbolique), /etc/sysconfig, /lib/services, /lib/lsb (lien symbolique)

Descriptions courtes

checkfs Vérifie l'intégrité des systèmes de fichiers avant leur montage (à l'exception des systèmes de fichiers journalisés ou réseau).

cleanfs Supprime les fichiers qui ne devraient pas être conservés entre deux redémarrages, tels que ceux dans `/run/` et `/var/lock/` ; il recrée `/run/utmp` et supprime les fichiers `/etc/nologin`, `/fastboot` et `/forcefsck` s'ils sont présents

console Charge la bonne table de correspondance du clavier ; initialise aussi la police d'écran

functions Contient des fonctions communes, telles que la vérification d'erreurs et d'états, utilisées par plusieurs scripts de démarrage

halt Arrête le système

ifdown Arrête un périphérique réseau

ifup Initialise un périphérique réseau

localnet Configure le nom d'hôte du système et le périphérique de boucle locale

modules Charge les modules du noyau listés dans `/etc/sysconfig/modules`, en utilisant les arguments qui y sont donnés

mountfs Monte tous les systèmes de fichiers, sauf ceux marqués *noauto* ou les systèmes réseaux

mountvirtfs Monte les systèmes de fichiers virtuels fournis par le noyau, tels que `proc`

network Configure les interfaces réseaux, telles que les cartes réseaux, et configure la passerelle par défaut (le cas échéant).

rc Script de contrôle du niveau d'exécution maître ; il est responsable du lancement individuel des autres scripts de démarrage, selon une séquence déterminée par le nom des liens symboliques en cours de traitement

reboot Redémarre le système

sendsignals S'assure que chaque processus est terminé avant que le système ne redémarre ou s'arrête

setclock Réinitialise l'horloge noyau avec l'heure locale au cas où l'horloge matérielle n'est pas en temps UTC

ipv4-static Fournit les fonctionnalités nécessaires à l'affectation d'une adresse IP statique à une interface réseau

swap Active et désactive les fichiers d'échange et les partitions

sysctl Charge la configuration du système à partir de `/etc/sysctl.conf`, si ce fichier existe, dans le noyau en cours d'exécution

sysklogd Lance et arrête les démons des journaux système et noyau

template Un modèle pour créer des scripts de démarrage personnalisés pour d'autres démons

udev Prépare le répertoire `/dev` et lance Udev

udev_retry Réessaie les uevents udev échoués, et copie les fichiers de règles générés de `/run/udev` vers `/etc/udev/rules.d` si nécessaire

9.3. Manipulation des périphériques et modules

Au Chapitre 8, nous avons installé le paquet udev en construisant eudev . 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 existe ou non. Ceci était le plus souvent réalisé 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 sont créés à chaque lancement du système, ils sont 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 entendu 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`, `/usr/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

Lorsque 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 possède un alias spécifique au bus et que le pilote du bus envoie correctement les alias nécessaires vers `sysfs`. Autrement, il faut organiser le chargement des modules par d'autres moyens. Avec Linux-5.18.3, 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 ultérieurement.

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-5.18.3, c'est le cas pour les bus ISA. Attendez que ce problème soit corrigé 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 ».

Si le module en question n'est pas un emballage et s'avère utile en tant que tel, configurez le script de démarrage **modules** pour charger ce module sur le système de démarrage. Pour cela, ajoutez le nom du module au fichier `/etc/sysconfig/modules` sur une ligne séparée. Ceci fonctionne aussi pour les modules d'emballage, mais sans être optimal.

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 réalisé 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. Si ce n'est pas le cas, 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 cela 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 `/usr/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 9.5, « 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. Périphériques réseaux

Udev, par défaut, nomme les périphériques réseaux à partir des données du Firmware/BIOS ou de leurs caractéristiques physiques comme leur bus, leur slot ou leur adresse MAC. Le but de cette convention de nommage est de s'assurer que les périphériques réseaux sont nommés de façon cohérente et non en fonction de l'heure à laquelle la carte réseau a été trouvée. Par exemple, sur un ordinateur possédant deux cartes réseaux Intel et Realtek, il se peut que la carte réseau Intel soit nommée eth0 et la Realtek soit nommée eth1. Dans certains cas, au redémarrage, les cartes sont numérotées en sens inverse.

Avec la nouvelle règle de nommage, les noms des cartes réseaux ressembleraient en général à quelque chose comme enp5s0 ou wlp3s0. Si cette convention de nommage ne vous plaît pas, vous pouvez implémenter la convention de nommage traditionnelle ou une personnalisée.

9.4.1.1. Désactiver la conservation des noms en ligne de commandes du noyau

Vous pouvez rétablir la règle de nommage traditionnelle qui utilise eth0, eth1, etc en ajoutant **`net.ifnames=0`** à la ligne de commandes du noyau. Ceci est surtout adapté aux systèmes n'ayant qu'un périphérique ethernet du même type. Les portables ont souvent plusieurs ports ethernet appelés eth0 et wlan0, ils sont donc éligibles à cette méthode. La ligne de commandes se passe dans le fichier de configuration de GRUB. Voir Section 10.4.4, « Créer le fichier de configuration de GRUB ».

9.4.1.2. Créer des règles Udev personnalisées

Vous pouvez personnaliser les règles de nommage en créant des règles udev personnalisées. Un script est inclus pour générer les règles initiales. Générez ces règles en lançant :

```
bash /usr/lib/udev/init-net-rules.sh
```

Maintenant, lisez le fichier `/etc/udev/rules.d/70-persistent-net.rules` pour trouver le nom affecté à une carte réseau :

```
cat /etc/udev/rules.d/70-persistent-net.rules
```

Note

Dans certains cas, comme lorsqu'une adresse MAC est affectée manuellement à une carte réseau ou dans un environnement virtuel tel que Qemu ou Xen, il se peut que le fichier des règles du réseau n'ait pas été généré car les adresses ne sont pas affectées de façon cohérente. Dans ces cas-là, vous ne pouvez pas utiliser cette méthode.

Le fichier commence par un bloc en commentaire suivi de deux lignes pour chaque NIC. La première ligne d'un NIC est une description commentée indiquant ses ID matériels (comme ses ID PCI de fabricant et de périphérique si c'est une carte PCI), avec le pilote entre parenthèses, si le pilote est détecté. Ni l'ID matériel ni le pilote ne sont utilisés pour déterminer le nom à donner à une interface ; ces informations ne sont présentes qu'à titre informatif. La deuxième ligne est la règle udev correspondant à cette NIC et qui lui affecte un nom.

Toutes les règles udev se composent de clés séparées par des virgules et éventuellement des espaces. Les clés de cette règle et l'explication de chacune sont :

- `SUBSYSTEM=="net"` — Ceci dit à udev d'ignorer les périphériques n'étant pas des cartes réseaux.
- `ACTION=="add"` — Ceci dit à udev d'ignorer la règle pour un uevent autre qu'un ajout (les uevents « remove » et « change » se produisent aussi mais il n'est pas utile de renommer les interfaces réseaux).
- `DRIVERS=="?*"` — Ceci existe pour que udev ignore les sous-interfaces VLAN ou les ponts (ces interfaces n'ayant pas de pilote). Ces sous-interfaces sont ignorées car le nom qui leur serait affecté entrerait en conflit avec leurs périphériques parents.
- `ATTR{address}` - La valeur de cette clé est l'adresse MAC de la NIC.
- `ATTR{type}=="1"` — Ceci garantit que la règle ne corresponde qu'à l'interface primaire au cas où certains pilotes sans fil créent plusieurs interfaces virtuelles. Les interfaces secondaires sont ignorées pour la même raison que l'on évite les sous-interfaces VLAN et les ponts : il y aurait conflit de noms.
- `NAME` — La valeur de cette clé est le nom donné par udev à cette interface.

La valeur de `NAME` est la partie importante. Assurez-vous de connaître le nom affecté à chacune de vos cartes réseaux avant de continuer, et d'utiliser cette valeur `NAME` quand vous créez les fichiers de configuration ci-dessous.

9.4.2. Liens symboliques pour le CD-ROM

Certains logiciels que vous pourriez vouloir installer ultérieurement (comme divers lecteurs multimédias) s'attendent à ce que les liens symboliques `/dev/cdrom` et `/dev/dvd` existent et pointent vers le lecteur CD-ROM ou DVD-ROM. De plus, il peut être pratique d'insérer des références à ces liens symboliques dans `/etc/fstab`. Udev est fourni avec un script qui génèrera des fichiers de règles pour créer ces liens symboliques pour vous, selon les possibilités de chaque périphérique, mais vous devez décider lequel des deux modes opératoires vous souhaitez que le script utilise.

Tout d'abord, le script peut opérer en mode « chemin » (utilisé par défaut pour les périphériques USB et FireWire), où les règles qu'il crée dépendent du chemin physique vers le lecteur CD ou DVD. Ensuite, il peut opérer en mode « id » (par défaut pour les périphériques IDE et SCSI), où les règles qu'il crée dépendent des chaînes d'identification contenues dans le lecteur CD ou DVD lui-même. Le chemin est déterminé par le script **path_id** d'Udev, et les chaînes d'identification sont lues à partir du matériel par ses programmes **ata_id** ou **scsi_id**, selon le type de périphérique que vous possédez.

Il existe des avantages pour chaque approche ; la bonne approche à utiliser dépendra des types de changements de périphérique qui peuvent se produire. Si vous vous attendez à ce que le chemin physique vers le périphérique (c'est-à-dire, les ports ou les fentes par lesquels ils sont branchés) changent, par exemple parce que vous envisagez de déplacer le lecteur sur un port IDE différent ou un connecteur USB différent, alors vous devriez utiliser le mode « id ». D'un autre côté, si vous vous attendez à ce que l'identification du périphérique change, par exemple parce qu'il peut mourir, et que vous le remplaciez par un périphérique différent avec les mêmes capacités et qui serait monté sur les mêmes connecteurs, vous devriez utiliser le mode « chemin ».

Si les deux types de changement sont possibles avec votre lecteur, choisissez un mode basé sur le type de changement que vous pensez rencontrer le plus fréquemment.

Important

Les périphériques externes (par exemple un lecteur CD connecté en USB) ne devraient pas utiliser la méthode des chemins, car chaque fois que le périphérique est monté sur un nouveau port, son chemin physique change. Tous les périphériques connectés en externe auront ce problème si vous écrivez des règles udev pour les reconnaître par leur chemin physique ; le problème ne concerne pas que les lecteurs CD et DVD.

Si vous souhaitez voir les valeurs que les scripts udev utiliseront, alors, pour celles appropriées au périphérique CD-ROM, trouvez le répertoire correspondant sous `/sys` (cela peut être par exemple `/sys/block/hdd`) et lancez une commande ressemblant à ce qui suit :

```
udevadm test /sys/block/hdd
```

Regardez les lignes contenant la sortie des divers programmes *_id. Le mode « id » utilisera la valeur ID_SERIAL si elle existe et n'est pas vide, sinon il utilisera une combinaison d'ID_MODEL et d'ID_REVISION. Le mode « chemin » utilisera la valeur d'ID_PATH.

Si le mode par défaut ne convient pas à votre situation, vous pouvez faire la modification suivante du fichier `/etc/udev/rules.d/83-cdrom-symlinks.rules`, comme suit, (où *mode* est soit « by-id » soit « by-path ») :

```
sed -e 's/"write_cd_rules"/"write_cd_rules mode"/' \
    -i /etc/udev/rules.d/83-cdrom-symlinks.rules
```

Remarquez qu'il n'est pas nécessaire de créer les fichiers de règles ou les liens symboliques à ce moment puisque vous avez monté en bind le répertoire `/dev` du système hôte dans le système LFS, et nous supposons que les liens symboliques existent sur l'hôte. Les règles et les liens symboliques seront créés la première fois que vous démarrerez votre système LFS.

Cependant, si vous avez plusieurs lecteurs CD-ROM, les liens symboliques générés à ce moment peuvent pointer vers des périphériques différents de ceux vers lesquels ils pointent sur votre hôte, car les périphériques ne sont pas découverts dans un ordre prévisible. Les affectations créées quand vous démarrerez pour la première fois le système LFS seront stables, donc cela n'est un problème que si vous avez besoin que les liens symboliques sur les deux systèmes pointent vers le même périphérique. Si tel est le cas, inspectez (et éditez peut-être) le fichier `/etc/udev/rules.d/70-persistent-cd.rules` généré après le démarrage pour vous assurer que les liens symboliques affectés correspondent à ce dont vous avez besoin.

9.4.3. 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 9.5, « 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+="webcam"
KERNEL=="video*", ATTRS{device}=="0x036f",  ATTRS{vendor}=="0x109e", SYMLINK+="tvtuner"

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 certains des liens symboliques `/dev/tvtuner` et `/dev/webcam` pointent toujours vers le bon périphérique.

9.5. Configuration générale du réseau

9.5.1. Création des fichiers de configuration d'interface réseau

Les interfaces activées et désactivées par le script réseau dépendent normalement des fichiers du répertoire `/etc/sysconfig/`. Ce répertoire devrait contenir un fichier par interface à configurer, tel que `ifconfig.xyz`, où « xyz » doit décrire la carte réseau. En général le nom d'interface (comme eth0) est suffisant. Dans ce fichier, se trouvent les attributs de cette interface, tels que son//ses adresse(s) IP, les masques de sous-réseau, etc. Il faut que le fichier ait pour nom *ifconfig*.

Note

Si vous n'avez pas suivi la procédure de la section précédente, udev affectera un nom à l'interface de carte réseau en se basant sur les caractéristiques physiques du système comme enp2s1. Si vous n'êtes pas sûr du nom de votre interface, vous pouvez toujours lancer **ip link** ou **ls /sys/class/net** après avoir démarré votre système.

Les noms d'interface dépendent de l'implémentation et de la configuration du démon udev exécuté sur le système. Le démon udev de LFS (installé au Section 8.70, « Eudev-3.2.11 ») ne sera pas exécuté à moins de démarrer le système LFS. Il n'est donc pas fiable de déterminer les noms d'interface utilisés dans le système LFS en exécutant ces commandes sur la distribution hôte, *même dans l'environnement chroot*.

La commande suivante crée un fichier modèle pour le périphérique *eth0* avec une adresse IP statique :

```
cd /etc/sysconfig/
cat > ifconfig.eth0 << "EOF"
ONBOOT=yes
IFACE=eth0
SERVICE=ipv4-static
IP=192.168.1.2
GATEWAY=192.168.1.1
PREFIX=24
BROADCAST=192.168.1.255
EOF
```

Les valeurs en italiques doivent être modifiées dans chaque fichier pour correspondre à la bonne configuration.

Si la variable `ONBOOT` est configurée à « yes », le script réseau de System V configurera la carte d'interface réseau (NIC) pendant le démarrage du système. S'il est configuré avec toute autre valeur que « yes », le NIC sera ignoré par le script réseau et ne sera pas configuré automatiquement. On peut démarrer et arrêter l'interface à la main avec les commandes **ifup** et **ifdown**.

La variable `IFACE` définit le nom de l'interface, par exemple, eth0. Elle est nécessaire dans tous les fichiers de configuration des périphériques réseaux. L'extension des fichiers doit correspondre à cette valeur.

La variable SERVICE définit la méthode utilisée pour obtenir l'adresse IP. Les scripts de démarrage LFS ont un format d'affectation d'IP modulaire. Créer les fichiers supplémentaires dans le répertoire /lib/services/ autorise d'autres méthodes d'affectation d'IP. Ceci est habituellement utilisé pour le DHCP, qui est adressé dans le livre BLFS.

La variable GATEWAY devrait contenir l'adresse IP par défaut de la passerelle, si elle existe. Sinon, mettez entièrement en commentaire la variable.

La variable PREFIX contient le nombre de bits utilisés dans le sous-réseau. Chaque octet dans une adresse IP est exprimé sur huit bits. Si le masque du sous-réseau est 255.255.255.0, alors il est en train d'utiliser les trois premiers octets (24 bits) pour spécifier le numéro du réseau. Si le masque réseau est 255.255.255.240, il utiliserait les 28 premiers bits. Les préfixes plus longs que 24 bits sont habituellement utilisés par les fournisseurs d'accès Internet ADSL et câble. Dans cet exemple (PREFIX=24), le masque réseau est 255.255.255.0. Ajustez la variable PREFIX en concordance avec votre sous-réseau spécifique. Si vous ne le mettez pas, PREFIX vaut 24 par défaut.

Pour plus d'informations, voir la page de manuel de **ifup**.

9.5.2. Créer le fichier /etc/resolv.conf

Le système aura besoin d'un moyen pour obtenir un résolveur 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. Créez ce fichier en lançant :

```
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 la 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.

Note

Les adresses des DNS publiques IPV4 de Google sont 8.8.8.8 et 8.8.4.4.

9.5.3. 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.

9.5.4. Personnaliser le fichier /etc/hosts

Choisissez l'adresse IP, son nom de domaine pleinement qualifié (*fully-qualified domain name*, ou FQDN) et les alias possibles à déclarer dans le fichier /etc/hosts. La syntaxe 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 en ont besoin pour fonctionner correctement.

Créez le fichier `/etc/hosts` en lançant :

```
cat > /etc/hosts << "EOF"
# Begin /etc/hosts

127.0.0.1 localhost.localdomain localhost
127.0.1.1 <FQDN> <HOSTNAME>
<192.168.1.1> <FQDN> <HOSTNAME> [alias1] [alias2 ...]
::1       localhost ip6-localhost ip6-loopback
ff02::1   ip6-allnodes
ff02::2   ip6-allrouters

# End /etc/hosts
EOF
```

Les valeurs *<192.168.1.1>*, *<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.

9.6. Utiliser et configurer les scripts de démarrage de System V

9.6.1. Comment fonctionnent les scripts de démarrage de System V ?

Linux utilise un service de démarrage spécial nommé SysVinit qui est basé sur un concept de *niveaux d'exécution*. Il peut être très différent d'un système à un autre, c'est pourquoi il ne peut pas être supposé que si cela fonctionne sur une distribution Linux particulière, cela fonctionnera de la même façon dans LFS. LFS a sa propre façon de faire les choses mais il respecte généralement les standards établis.

SysVinit (which will be referred to as « init » from now on) works using a run-levels scheme. There are seven (numbered 0 to 6) run-levels (actually, there are more run-levels, but they are for special cases and are generally not used. See `init(8)` for more details), and each one of those corresponds to the actions the computer is supposed to perform when it starts up. The default run-level is 3. Here are the descriptions of the different run-levels as they are implemented in LFS:

```
0: halt the computer
1: single-user mode
2: reserved for customization, otherwise does the same as 3
3: multi-user mode with networking
4: reserved for customization, otherwise does the same as 3
5: same as 4, it is usually used for GUI login (like GNOME's gdm or LXDE's lxdm)
6: reboot the computer
```

Note

Classically, run level 2 above was defined as "multi-user mode without networking", but this was only the case many years ago when multiple users could log into a system connected via serial ports. In today's environment it makes no sense and we designate it now as "reserved".

9.6.2. Configuration de Sysvinit

Lors de l'initialisation du noyau, le premier programme qui se lance est soit spécifié sur la ligne de commande, soit, par défaut, **init**. Ce programme lit le fichier d'initialisation `/etc/inittab`. Créez ce fichier avec :

```
cat > /etc/inittab << "EOF"
# Begin /etc/inittab

id:3:initdefault:

si::sysinit:/etc/rc.d/init.d/rc S

l0:0:wait:/etc/rc.d/init.d/rc 0
l1:S1:wait:/etc/rc.d/init.d/rc 1
l2:2:wait:/etc/rc.d/init.d/rc 2
l3:3:wait:/etc/rc.d/init.d/rc 3
l4:4:wait:/etc/rc.d/init.d/rc 4
l5:5:wait:/etc/rc.d/init.d/rc 5
l6:6:wait:/etc/rc.d/init.d/rc 6

ca:12345:ctrlaltdel:/sbin/shutdown -t1 -a -r now

su:S06:once:/sbin/sulogin
s1:1:respawn:/sbin/sulogin

1:2345:respawn:/sbin/agetty --noclear tty1 9600
2:2345:respawn:/sbin/agetty tty2 9600
3:2345:respawn:/sbin/agetty tty3 9600
4:2345:respawn:/sbin/agetty tty4 9600
5:2345:respawn:/sbin/agetty tty5 9600
6:2345:respawn:/sbin/agetty tty6 9600

# End /etc/inittab
EOF
```

Vous trouverez une explication de ce fichier d'initialisation dans la page de manuel de *inittab*. Pour LFS, la commande clé qui se lance est **rc**. Le fichier d'initialisation ci-dessus demandera à **rc** de lancer tous les scripts commençant par un S qui sont dans le répertoire `/etc/rc.d/rcS.d`, puis tous les scripts commençant par un S du répertoire `/etc/rc.d/rc?.d` où le point d'interrogation est spécifié par la valeur initdefault.

Par commodité, le script **rc** lit une bibliothèque de fonctions dans `/lib/lsb/init-functions`. Cette bibliothèque lit aussi un fichier de configuration facultatif, `/etc/sysconfig/rc.site`. Tous les paramètres du fichier de configuration du système décrits dans les sections suivantes peuvent être mis dans ce fichier, ce qui permet de rassembler tous les paramètres systèmes dans un seul fichier.

Pour faciliter le débogage, le script functions enregistre aussi toute la sortie dans `/run/var/bootlog`. Le répertoire `/run` étant un tmpfs, ce fichier n'est pas persistant entre les redémarrages, il est cependant envoyé dans le fichier plus permanent `/var/log/boot.log` à la fin du processus de démarrage.

9.6.2.1. Modifier les niveaux d'exécution

La commande utilisée pour modifier le niveau d'exécution est **init** ***<[niveau_exécution]>***, où *<[niveau_exécution]>* est le niveau d'exécution cible. Par exemple, pour redémarrer l'ordinateur, un utilisateur pourrait lancer la commande **init 6** qui est un alias de la commande **reboot**. De même, **init 0** est un alias pour la commande **halt**.

There are a number of directories under `/etc/rc.d` that look like `rc?.d` (where ? is the number of the run-level) and `rcS.d`, all containing a number of symbolic links. Some begin with a *K*, the others begin with an *S*, and all of them have two numbers following the initial letter. The K means to stop (kill) a service and the S means to start a service. The numbers determine the order in which the scripts are run, from 00 to 99—the lower the number the earlier it gets executed. When **init** switches to another run-level, the appropriate services are either started or stopped, depending on the runlevel chosen.

Les vrais scripts sont dans `/etc/rc.d/init.d`. Ce sont eux qui font le vrai travail et les liens symboliques pointent tous vers eux. Les liens K et les liens S pointent vers le même script dans `/etc/rc.d/init.d`. Ceci est dû au fait que les scripts peuvent être appelés avec différents paramètres comme *`start`*, *`stop`*, *`restart`*, *`reload`* et *`status`*. Quand un lien K est rencontré, le script approprié est lancé avec l'argument *`stop`*. Quand un lien S est rencontré, le script approprié est lancé avec l'argument *`start`*.

Voici les descriptions de ce que font les arguments des scripts :

`start`
: Le service est lancé.

`stop`
: Le service est arrêté.

`restart`
: Le service est arrêté puis relancé.

`reload`
: La configuration du service est mise à jour. Cette commande est utilisée après la modification du fichier de configuration d'un service, quand le service n'a pas besoin d'être redémarré.

`status`
: Indique si le service est en cours d'exécution ainsi que les PID associés.

Vous êtes libre de modifier la façon dont le processus de démarrage fonctionne : après tout, c'est votre système LFS. Les fichiers donnés le sont à titre d'exemples.

9.6.3. Les scripts de démarrage Udev

Le script de démarrage `/etc/rc.d/init.d/udev` lance **udevd**, récupère les périphériques « branchés à froid » créés d'ores et déjà par le noyau et attend des règles pour se terminer. Le script supprime aussi du gestionnaire d'uevent le réglage par défaut `/sbin/hotplug` . On fait cela car le noyau n'a plus besoin de faire appel à un binaire externe. Par contre, **udevd** va écouter sur un socket netlink les uevents engendrés par le noyau.

Le script de démarrage **/etc/rc.d/init.d/udev_retry** se charge de récupérer les événements des sous-systèmes dont les règles s'appuient sur des systèmes de fichiers non montés jusqu'à ce que le script **mountfs**soit lancé (en particulier, `/usr` et `/var` peuvent avoir cet effet). Ce script s'exécute après le script **mountfs**, donc ces règles (si elles sont de nouveau récupérées) devraient s'appliquer la deuxième fois. Il se configure à partir du fichier `/etc/sysconfig/udev_retry` ; donc tout mot autre que des commentaires dans ce fichier est vu comme un nom de sous-système à rattraper lorsqu'il lance le nouvel essai. Pour trouver le sous-système d'un périphérique, utilisez **udevadm info --attribute-walk <périphérique>**, où <périphérique> est un chemin absolu dans /dev ou /sys comme /dev/sr0 ou /sys/class/rtc.

Pour plus d'informations sur le chargement des modules du noyau et udev, consultez Section 9.3.2.3, « Chargement d'un module ».

9.6.4. Configurer l'horloge du système

Le script **setclock** lit l'heure à partir de l'horloge matérielle, appelée aussi horloge BIOS ou *Complementary Metal Oxide Semiconductor* (CMOS). Si l'horloge matérielle est réglée sur UTC, ce script convertira l'heure de l'horloge matérielle en heure locale en utilisant le fichier `/etc/localtime` (qui indique au programme **hwclock** le fuseau horaire de l'utilisateur). Il n'y a aucun moyen de détecter si l'horloge matérielle est réglée sur UTC, donc vous devez configurer ce paramètre manuellement.

Le script **setclock** se lance via udev quand le noyau détecte le matériel au démarrage. Vous pouvez aussi le lancer manuellement avec le paramètre stop pour stocker l'heure du système dans l'horloge CMOS.

Si vous ne vous rappelez pas si l'horloge matérielle est réglée sur UTC, assurez-vous-en en lançant la commande **`hwclock --localtime --show`**. Cela affichera l'heure actuelle selon l'horloge matérielle. Si elle correspond à ce qu'indique votre montre, l'horloge matérielle est en heure locale. Si la sortie de **hwclock** n'est pas l'heure locale, il y a des chances qu'il s'agisse de l'heure UTC. Vérifiez-le en ajoutant ou enlevant le bon nombre d'heures du fuseau horaire de l'heure affichée avec **hwclock**. Par exemple, si vous êtes dans le fuseau MST, connu aussi sous le nom GMT -0700, ajoutez sept heures à l'heure locale.

Changez la valeur de la variable `UTC` ci-dessous en *`0`* (zéro) si l'horloge matérielle *n'est pas* réglée sur l'heure UTC.

Créez un nouveau fichier `/etc/sysconfig/clock` en lançant ce qui suit :

```
cat > /etc/sysconfig/clock << "EOF"
# Début de /etc/sysconfig/clock

UTC=1

# Mettez ici les options que vous pourriez avoir besoin de donner à hwclock,
# comme le type de l'horloge matérielle de la machine pour les Alphas.
CLOCKPARAMS=

# Fin de /etc/sysconfig/clock
EOF
```

Une bonne astuce qui explique la gestion de l'heure sur LFS est disponible sur *http://www.fr.linuxfromscratch.org/view/astuces/heure.txt*. Elle traite de sujets tels que les fuseaux horaires, UTC et la variable d'environnement `TZ`.

Note

Vous pouvez aussi régler les paramètres CLOCKPARAMS et UTC dans le fichier `/etc/sysconfig/rc.site`.

9.6.5. Configurer la Console Linux

Cette section traite de la configuration du script de démarrage **console**, qui initialise la disposition du clavier, la police de la console et le niveau de journalisation du noyau. Si vous n'utilisez pas les caractères non-ASCII (par exemple le symbole du copyright, de la livre sterling et de l'euro) et que le clavier est américain, la plupart de cette section peut être sautée. Sans ce fichier de configuration, (ou des options équivalentes dans `rc.site`), le script de démarrage **console** ne fera rien.

Le script **console** lit les informations de configuration du fichier `/etc/sysconfig/console`. Il décide de la disposition de clavier et de la police de la console à utiliser. Différents guides pratiques spécifiques aux langues peuvent aussi être d'une grande aide (voir *http://www.tldp.org/HOWTO/HOWTO-INDEX/other-lang.html*). Si vous avez toujours des doutes, jetez un œil dans les répertoires `/usr/share/keymaps` et `/usr/share/consolefonts` pour des dispositions de clavier valides et des polices d'écran. Lisez les pages de manuel `loadkeys(1)` et `setfont(8)` afin de déterminer les bons arguments pour ces programmes.

Le fichier `/etc/sysconfig/console` devrait contenir des lignes de la forme : VARIABLE="valeur". Les variables suivantes sont reconnues :

LOGLEVEL

Cette variable spécifie le niveau de traçage pour les messages du noyau envoyés à la console, selon le paramétrage de **dmesg -n**. Les niveaux valides vont de « 1 » (aucun message) à « 8 ». Le niveau par défaut est « 7 ».

KEYMAP

Cette variable spécifie les arguments du programme **loadkeys**, en général, le nom de l'arrangement du clavier à charger, comme « it ». Si cette variable n'est pas initialisée, le script de démarrage ne lancera pas le programme **loadkeys** et l'arrangement par défaut du noyau sera utilisé. Remarquez qu'un petit nombre d'arrangements ont plusieurs versions avec le même nom (cz avec ses variantes dans qwerty/ et qwertz/, es dans olpc/ et qwerty/, et trf dans fgGlod/ et qwerty/). Dans ce cas, le répertoire parent doit être spécifié (par exemple qwerty/es) pour s'assurer que l'arrangement adéquat est chargé.

KEYMAP_CORRECTIONS

Cette variable (rarement utilisée) spécifie les arguments du second appel au programme **loadkeys**. Elle est utile si la disposition du clavier par défaut n'est pas totalement satisfaisante et que vous devez faire un petit ajustement. Par exemple, pour inclure le symbole euro dans une disposition de clavier qui ne l'a normalement pas, réglez cette variable à « euro2 ».

FONT

Cette variable spécifie les arguments du programme **setfont**. En principe, cela inclut le nom de la police, « -m » et le nom de la disposition de clavier de l'application à charger. Par exemple, pour charger la police « lat1-16 » avec la disposition de clavier de l'application « 8859-1 », (comme il convient aux États-Unis), réglez cette variable à « lat1-16 -m 8859-1 ». En mode UTF-8, le noyau utilise la disposition de clavier de l'application pour la conversion de codes de touche 8-bits composés dans la disposition de clavier en UTF-8, et ainsi vous devriez initialiser l'argument du paramètre "-m" à l'encodage des codes de touche composés dans la disposition de clavier.

UNICODE

Réglez cette variable à « 1 », « yes » ou « true » afin de mettre la console en mode UTF-8. Ceci est utile pour les paramètres linguistiques basés sur UTF-8 mais nuisible dans le cas contraire.

LEGACY_CHARSET

Pour beaucoup de types de clavier, il n'y a pas de disposition de clavier pour le stock Unicode dans le paquet Kbd. Le script de démarrage **console** convertira une disposition de clavier disponible en UTF-8 au vol si cette variable est réglée à l'encodage de la disposition du clavier non UTF-8 disponible.

Quelques exemples :

- Pour une initialisation non Unicode, en général seules les variables KEYMAP et FONT sont nécessaires. Par exemple, pour l'initialisation en polonais, on utiliserait :

```
cat > /etc/sysconfig/console << "EOF"
# Début de /etc/sysconfig/console

KEYMAP="pl2"
FONT="lat2a-16 -m 8859-2"

# Fin de /etc/sysconfig/console
EOF
```

- Comme mentionné ci-dessus, il est parfois nécessaire d'ajuster légèrement une disposition de clavier stockée. L'exemple suivant ajoute le symbole euro a la disposition allemande du clavier :

```
cat > /etc/sysconfig/console << "EOF"
# Begin /etc/sysconfig/console

KEYMAP="de-latin1"
KEYMAP_CORRECTIONS="euro2"
FONT="lat0-16 -m 8859-15"
UNICODE="1"

# Fin de /etc/sysconfig/console
EOF
```

- Ce qui suit est un exemple où l'Unicode est activé pour le bulgare, et où une disposition du clavier UTF-8 par défaut existe :

```
cat > /etc/sysconfig/console << "EOF"
# Début de /etc/sysconfig/console

UNICODE="1"
KEYMAP="bg_bds-utf8"
FONT="LatArCyrHeb-16"

# Fin de /etc/sysconfig/console
EOF
```

- Du fait de l'utilisation d'une police 512 glyphes LatArCyrHeb-16 dans l'exemple précédent, les couleurs claires ne sont plus disponibles sur la console Linux à moins d'utiliser un framebuffer. Si vous voulez avoir les couleurs claires sans framebuffer et que vous n'avez pas forcément besoin des caractères n'appartenant pas à votre langue, il est encore possible d'utiliser une police 256 glyphes spécifique à votre langue, comme illustré ci-dessous :

```
cat > /etc/sysconfig/console << "EOF"
# Début de /etc/sysconfig/console

UNICODE="1"
KEYMAP="bg_bds-utf8"
FONT="cyr-sun16"

# Fin de /etc/sysconfig/console
EOF
```

- L'exemple suivant illustre l'auto-conversion de la disposition de clavier d'ISO-8859-15 vers UTF-8 et l'activation des touches mortes en mode Unicode :

```
cat > /etc/sysconfig/console << "EOF"
# Début de /etc/sysconfig/console

UNICODE="1"
KEYMAP="de-latin1"
KEYMAP_CORRECTIONS="euro2"
LEGACY_CHARSET="iso-8859-15"
FONT="LatArCyrHeb-16 -m 8859-15"

# Fin de /etc/sysconfig/console
EOF
```

- Certaines dispositions de codage comportent des touches mortes (c-à-d que les touches ne produisent pas un caractère en elles-mêmes, mais mettent un accent sur le caractère produit par la touche suivante) ou définissent des règles de comportement (comme : « Appuyez sur Ctrl+. A E pour obtenir Æ » dans la disposition du clavier par défaut). Linux-5.18.3 n'interprète correctement les touches mortes et les règles de composition que quand les caractères sources qui seront composés ensemble sont multi-octet. Ce défaut n'affecte pas les dispositions de clavier pour les langues européennes, car les accents sont ajoutés à des caractères ASCII non accentués, ou

deux caractères ASCII sont composés ensemble. Néanmoins en mode UTF-8, c'est un problème, comme pour la langue grecque, où on a parfois besoin de mettre un accent sur la lettre « alpha ». La solution consiste soit à éviter d'utiliser UTF-8, soit à installer le système de fenêtrage X qui n'a pas cette limitation dans sa gestion de l'entrée.

- Pour le chinois, le japonais, le coréen et certaines autres langues, la console Linux ne peut pas être configurée pour afficher les caractères nécessaires. Les utilisateurs qui ont besoin de telles langues doivent installer le système de fenêtrage X, dont les polices couvrent la plage de caractères nécessaire et qui a la bonne méthode d'entrée (par exemple, SCIM prend en charge une large variété de langues).

Note

Le fichier `/etc/sysconfig/console` ne contrôle que la localisation de la console texte de Linux. Cela n'a rien à voir avec le bon paramétrage du type de clavier et des polices du terminal dans le système de fenêtrage X, avec les sessions ssh ou une console série. Dans de telles situations, les limites mentionnées dans les deux derniers points de la liste ci-dessus ne s'appliquent pas.

9.6.6. Créer des fichiers au démarrage

Parfois, on veut créer des fichiers lors du démarrage. Par exemple, le répertoire `/tmp/.ICE-unix` est souvent requis. Vous pouvez le faire en créant une entrée dans le script de configuration `/etc/sysconfig/createfiles`. Le format de ce fichier est indiqué dans les commentaires du fichier de configuration par défaut.

9.6.7. Configurer le script sysklogd

Le script `sysklogd` invoque le programme **syslogd** faisant partie de l'initialisation par System V. L'option *`-m 0`* désactive la marque périodique que **syslogd** écrit par défaut dans les fichiers journaux toutes les 20 minutes. Si vous voulez activer cet horodatage, éditez `/etc/sysconfig/rc.site` et définissez la variable SYSKLOGD_PARMS à la valeur désirée. Par exemple, pour supprimer tous les paramètres, réglez la variable à la valeur null :

```
SYSKLOGD_PARMS=
```

Voir **`man syslogd`** pour plus d'options.

9.6.8. Le fichier rc.site

Le fichier facultatif `/etc/sysconfig/rc.site` contient les paramètres réglés automatiquement pour chaque script de démarrage de System V. Il peut aussi contrôler les valeurs des fichiers `hostname`, `console` et `clock` du répertoire `/etc/sysconfig/`. Si les variables associées se trouvent à la fois dans ces fichiers distincts et dans `rc.site`, les valeurs des fichiers spécifiques sont prioritaires.

`rc.site` contient aussi des paramètres pour personnaliser d'autres aspects du processus de démarrage. Le réglage de la variable IPROMPT permettra un lancement sélectif des scripts de démarrage. D'autres options sont décrites dans les commentaires du fichier. La version par défaut du fichier est ci-dessous :

```
# rc.site
# Optional parameters for boot scripts.

# Distro Information
# These values, if specified here, override the defaults
#DISTRO="Linux From Scratch" # The distro name
#DISTRO_CONTACT="lfs-dev@lists.linuxfromscratch.org" # Bug report address
#DISTRO_MINI="LFS" # Short name used in filenames for distro config

# Define custom colors used in messages printed to the screen

# Please consult `man console_codes` for more information
# under the "ECMA-48 Set Graphics Rendition" section
#
```

```
# Warning: when switching from a 8bit to a 9bit font,
# the linux console will reinterpret the bold (1;) to
# the top 256 glyphs of the 9bit font.  This does
# not affect framebuffer consoles

# These values, if specified here, override the defaults
#BRACKET="\\033[1;34m" # Blue
#FAILURE="\\033[1;31m" # Red
#INFO="\\033[1;36m"    # Cyan
#NORMAL="\\033[0;39m"  # Grey
#SUCCESS="\\033[1;32m" # Green
#WARNING="\\033[1;33m" # Yellow

# Use a colored prefix
# These values, if specified here, override the defaults
#BMPREFIX="      "
#SUCCESS_PREFIX="${SUCCESS}  *  ${NORMAL} "
#FAILURE_PREFIX="${FAILURE}*****${NORMAL} "
#WARNING_PREFIX="${WARNING} *** ${NORMAL} "

# Manually seet the right edge of message output (characters)
# Useful when resetting console font during boot to override
# automatic screen width detection
#COLUMNS=120

# Interactive startup
#IPROMPT="yes" # Whether to display the interactive boot prompt
#itime="3"    # The amount of time (in seconds) to display the prompt

# The total length of the distro welcome string, without escape codes
#wlen=$(echo "Welcome to ${DISTRO}" | wc -c )
#welcome_message="Welcome to ${INFO}${DISTRO}${NORMAL}"

# The total length of the interactive string, without escape codes
#ilen=$(echo "Press 'I' to enter interactive startup" | wc -c )
#i_message="Press '${FAILURE}I${NORMAL}' to enter interactive startup"

# Set scripts to skip the file system check on reboot
#FASTBOOT=yes

# Skip reading from the console
#HEADLESS=yes

# Write out fsck progress if yes
#VERBOSE_FSCK=no

# Speed up boot without waiting for settle in udev
#OMIT_UDEV_SETTLE=y

# Speed up boot without waiting for settle in udev_retry
#OMIT_UDEV_RETRY_SETTLE=yes

# Skip cleaning /tmp if yes
#SKIPTMPCLEAN=no

# For setclock
#UTC=1
#CLOCKPARAMS=

# For consolelog (Note that the default, 7=debug, is noisy)
#LOGLEVEL=7

# For network
#HOSTNAME=mylfs

# Delay between TERM and KILL signals at shutdown
#KILLDELAY=3
```

```
# Optional sysklogd parameters
#SYSKLOGD_PARMS="-m 0"

# Console parameters
#UNICODE=1
#KEYMAP="de-latin1"
#KEYMAP_CORRECTIONS="euro2"
#FONT="lat0-16 -m 8859-15"
#LEGACY_CHARSET=
```

9.6.8.1. Personnaliser les scripts de démarrage et d'extinction

Les scripts de démarrage LFS démarrent et arrêtent un système d'une façon très efficace, mais vous pouvez faire quelques modifications dans le fichier rc.site pour améliorer encore davantage la vitesse et ajuster les messages selon vos préférences. Pour cela, ajustez les paramètres du fichier `/etc/sysconfig/rc.site` ci-dessus.

- Pendant le script de démarrage `udev`, un appel à **udev settle** demande du temps pour s'achever. Ce temps peut être ou non nécessaire pour des périphériques présents dans votre système. Si vous n'avez que des partitions simples et une seule carte Ethernet, le processus de démarrage n'aura probablement pas besoin d'attendre cette commande. Pour la passer, définissez la variable OMIT_UDEV_SETTLE=y.
- Le script de démarrage `udev_retry` lance aussi par défaut **udev settle**. Cette commande n'est nécessaire par défaut que si le répertoire `/var` est monté séparément. En effet, la vérification a besoin du fichier `/var/lib/hwclock/adjtime`. D'autres personnalisations peuvent nécessiter d'attendre qu'udev se termine mais dans beaucoup d'installations, ce n'est pas nécessaire. Sautez la commande en définissant la variable OMIT_UDEV_RETRY_SETTLE=y.
- Par défaut, les vérifications des systèmes de fichiers sont sans message. Cela peut être vu comme un retard pendant le processus de démarrage. Pour activer la sortie de **fsck**, définissez la variable VERBOSE_FSCK=y.
- Lors du redémarrage, il se peut que vous vouliez passer complètement la vérification du système de fichiers, **fsck**. Vous pouvez pour cela soit créer le fichier `/fastboot`, soit redémarrer le système avec la commande **/sbin/shutdown -f -r now**. Inversement, vous pouvez forcer la vérification de tous les systèmes de fichiers en créant `/forcefsck` ou en lançant **shutdown** avec le paramètre `-F` plutôt que `-f`.

  La définition de la variable FASTBOOT=y désactivera **fsck** lors du processus de démarrage jusqu'à ce qu'il soit supprimé. Ce n'est pas recommandé de façon permanente.
- En principe, tous les fichiers du répertoire `/tmp` sont effacés au moment du démarrage. Selon le nombre de fichiers ou de répertoires présents, cela peut provoquer un retard important dans le processus de démarrage. Pour passer la suppression de ces fichiers, définissez la variable SKIPTMPCLEAN=y.
- Lors de l'arrêt, le programme **init** envoie un signal TERM à chaque programme qu'il a démarré (comme agetty), il attend un moment de définition (par défaut, 3 secondes), et il envoie à chaque processus un signal KILL puis attend de nouveau. Ce processus se répète dans le script **sendsignals** pour tous les processus non terminés par leurs propres scripts. Le délai de **init** peut être défini en passant un paramètre. Par exemple, pour supprimer le délai dans **init**, passez le paramètre -t0 lors de l'extinction ou du redémarrage (comme **/sbin/shutdown -t0 -r now**). Le délai du script **sendsignals** peut être passé en définissant le paramètre KILLDELAY=0.

9.7. Fichiers de démarrage du shell Bash

Le programme shell **/bin/bash** (dénommé ci-après « le shell ») utilise une collection de fichiers de démarrage pour aider à la création d'un environnement d'exécution. Chaque fichier a une utilisation spécifique et pourrait avoir des effets différents sur les environnements de connexion et interactif. Les fichiers du répertoire `/etc` fournissent un paramétrage global. Si un fichier équivalent existe dans le répertoire personnel, il pourrait surcharger les paramétrages globaux.

Un shell interactif de connexion est lancé après une connexion réussie, en utilisant **/bin/login**, par la lecture du fichier `/etc/passwd`. Un shell interactif sans connexion est lancé en ligne de commande (c'est-à-dire `[prompt]$` **/bin/bash**). Un shell non interactif est habituellement présent quand un script shell est en cours d'exécution. Il est non interactif parce qu'il traite un script et n'attend pas une saisie de l'utilisateur entre les commandes.

Pour plus d'informations, voir **info bash** sous la section *Bash Startup Files and Interactive Shells* (Fichiers de démarrage de Bash et shells interactifs).

Les fichiers `/etc/profile` et `~/.bash_profile` sont lus quand le shell est appelé en tant que shell interactif de connexion.

Le fichier `/etc/profile` de base ci-dessous configure quelques variables d'environnement nécessaires à la prise en charge native des langues. Les configurer convenablement permet ce qui suit :

- La sortie des programmes traduite dans la langue maternelle ;
- Un classement correct des caractères en lettres, chiffres et autres classes. Ceci est nécessaire pour que **bash** accepte correctement les caractères non-ASCII dans les lignes de commandes pour les locales autres que l'anglais ;
- L'ordre de tri alphabétique correct pour le pays ;
- Taille de papier par défaut appropriée
- Le bon formatage des valeurs monétaires, d'heure et de dates.

Remplacez *<ll>* ci-dessous par le code à deux lettres de la langue désirée (par exemple, « fr ») et *<CC>* avec le code à deux lettres du pays approprié (par exemple, « FR »). *<charmap>* devra être remplacé avec le jeu de caractères canonique de la locale choisie. Des modificateurs optionnels comme « @euro » peuvent aussi être présents.

La liste de toutes les locales prises en charge par Glibc peut être obtenue en exécutant la commande suivante :

```
locale -a
```

Les locales peuvent avoir plusieurs synonymes. Par exemple, « ISO-8859-1 » est aussi appelée « iso8859-1 » et « iso88591 ». Certaines applications ne peuvent pas gérer les différents synonymes correctement (elles nécessitent par exemple l'écriture de « UTF-8 » sous la forme « UTF-8 » et non « utf8 »), donc il est plus sûr de choisir le nom canonique pour une locale particulière. Pour déterminer le nom canonique, lancez la commande suivante, où *<nom locale>* est l'affichage donné par **locale -a** pour votre locale préférée (« fr_FR.iso88591 » dans notre exemple).

```
LC_ALL=<nom_de_la_locale> locale charmap
```

Pour la locale « fr_FR.iso88591 », la commande ci-dessus affichera :

```
ISO-8859-1
```

Ceci résulte en un paramétrage final de locale avec « fr_FR.ISO-8859-1 ». Il est important que la locale trouvée utilisant l'heuristique ci-dessus soit testée avant d'être ajoutée aux fichiers de démarrage de Bash :

```
LC_ALL=<nom_de_la_locale> locale language
LC_ALL=<nom_de_la_locale> locale charmap
LC_ALL=<nom_de_la_locale> locale int_curr_symbol
LC_ALL=<nom_de_la_locale> locale int_prefix
```

Les commandes ci-dessus devraient afficher le nom de la langue, le codage des caractères utilisé par le paramètre régional, la monnaie et le préfixe du pays à composer avant de saisir un numéro de téléphone. Si une des commandes ci-dessus échoue avec un message similaire à un de ceux montrés ci-dessous, cela signifie que votre paramètre linguistique n'a pas été installé au chapitre Section 8.5, « Glibc-2.35 » ou qu'il n'est pas pris 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 pouvez soit installer la locale désirée en utilisant la commande **localedef** soit considérer l'utilisation d'une locale différente. Les instructions suivantes supposent qu'il n'y a pas eu de tels messages de Glibc.

D'autres paquets peuvent aussi mal fonctionner (mais pourraient ne pas nécessairement afficher de messages d'erreurs) si le nom de la locale ne correspond pas à leur attente. Dans de tels cas, vous pouvez obtenir des informations utiles en cherchant comment les autres distributions Linux prennent en charge votre locale.

Une fois que les bons paramètres de locale ont été déterminés, créez le fichier `/etc/profile` :

```
cat > /etc/profile << "EOF"
# Début de /etc/profile

export LANG=<ll>_<CC>.<charmap><@modifiers>

# Fin de /etc/profile
EOF
```

Les paramètres régionaux « C » (par défaut) et « en_US.utf8 » (celle recommandée pour les utilisateurs de langue anglaise vivant aux États-Unis) sont différentes. « C » utilise le codage US-ASCII 7-bit et traite les octets utilisant le bit de poids le plus fort comme des caractères invalides. C'est pourquoi, par exemple, la commande **ls** les remplace par des points d'interrogation sous ce paramètre. De même, essayer d'envoyer un courriel avec de tels caractères depuis Mutt ou Pine envoi des messages non compatibles avec la RFC (le codage du mail sortant est indiqué comme « 8-bit inconnu »). Donc, vous ne pouvez utiliser le paramètre « C » que si vous êtes sûr de ne jamais avoir besoin de caractères 8-bit.

Les locales basées sur UTF-8 ne sont pas bien prises en charge par certains programmes. Le travail progresse pour documenter et, si possible, réparer de tels problèmes, voir *http://fr.linuxfromscratch.org/blfs/../view/blfs-svn/introduction/locale-issues.html*.

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"
# Begin /etc/inputrc
# Modified by Chris Lynn <roryo@roryo.dynup.net>

# Allow the command prompt to wrap to the next line
set horizontal-scroll-mode Off

# Enable 8-bit input
set meta-flag On
set input-meta On

# Turns off 8th bit stripping
set convert-meta Off

# Keep the 8th bit for display
set output-meta On

# none, visible or audible
set bell-style none

# All of the following map the escape sequence of the value
# contained in the 1st argument to the readline specific functions
"\eOd": backward-word
"\eOc": forward-word

# for linux console
"\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

# for xterm
"\eOH": beginning-of-line
"\eOF": end-of-line

# for Konsole
"\e[H": beginning-of-line
"\e[F": end-of-line

# End /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

# Fin de /etc/shells
EOF
```

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
proc           /proc        proc     nosuid,noexec,nodev 0     0
sysfs          /sys         sysfs    nosuid,noexec,nodev 0     0
devpts         /dev/pts     devpts   gid=5,mode=620      0     0
tmpfs          /run         tmpfs    defaults            0     0
devtmpfs       /dev         devtmpfs mode=0755,nosuid    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-5.18.3

Le paquet Linux contient le noyau Linux.

Temps de construction approximatif: 1.5 - 130.0 SBU (en gÃ©nÃ©ral environ 12 SBU)

Espace disque requis: 1200 - 8800 Mo (en gÃ©nÃ©ral environ 1700 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-svn/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 -->
   [ ] Compile the kernel with warnings as errors [CONFIG_WERROR]
   < > Enable kernel headers through /sys/kernel/kheaders.tar.xz [CONFIG_IKHEADERS]
Device Drivers  --->
  Graphics support --->
   Frame buffer Devices --->
      [*] Support for frame buffer devices ----
  Generic Driver Options  --->
   [ ] Support for uevent helper [CONFIG_UEVENT_HELPER]
   [*] Maintain a devtmpfs filesystem to mount at /dev [CONFIG_DEVTMPFS]
   [*]   Automount devtmpfs at /dev, after the kernel mounted the rootfs [CONFIG_DEVTMPFS_MOUNT]
```

Vous pourriez souhaiter d'autres options selon les besoins de votre système. Pour une liste des options nécessaires pour les paquets BLFS, voir *L'index des options du noyau pour BLFS* (http://fr.linuxfromscratch.org/blfs/../view/blfs-svn//longindex.html#kernel-config-index).

Note

Si votre matériel hôte utilise UEFI et que vous souhaitez démarrer LFS sans, vous devrez ajuster certaines configurations du noyau en suivant *la page BLFS*.

Voici pourquoi on vise les éléments de configuration ci-dessus :

`Compile the kernel with warnings as errors`
: This may cause building failure if the compiler and/or configuration are different from those of the kernel developers.

`Enable kernel headers through /sys/kernel/kheaders.tar.xz`
: Cela demandera **cpio** pour construire le noyau. **cpio** n'est pas installé par LFS.

`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.

`Automount devtmpfs at /dev`
: This will mount the kernel view of the devices on /dev upon switching to root filesystem just before starting init.

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-5.18.3` 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-5.18.3/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

If the host system has a separate /boot partition, the files copied below should go there. The easiest way to do that is to bind /boot on the host (outside chroot) to /mnt/lfs/boot before proceeding. As the `root` user in the *host system*:

```
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-5.18.3-lfs-r11.1-135+
```

`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-5.18.3
```

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-5.18.3
```

Installez la documentation du noyau Linux :

```
install -d /usr/share/doc/linux-5.18.3
cp -r Documentation/* /usr/share/doc/linux-5.18.3
```

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-5.18.3` 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-5.18.3 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-5.18.3, vmlinuz-5.18.3-lfs-r11.1-135+ et System.map-5.18.3
Répertoires installés: /lib/modules, /usr/share/doc/linux-5.18.3

Descriptions courtes

`config-5.18.3` Contient toutes les options de configuration choisies pour le noyau

`vmlinuz-5.18.3-lfs-r11.1-135+` 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-5.18.3` 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

Note

Si votre système prend en charge l'UEFI et que vous souhaitez démarrer LFS en UEFI, vous devriez sauter cette page, et configurer GRUB avec la prise en charge de l'UEFI avec les instructions de *la page BLFS*.

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
```

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 8. 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 5.18.3-lfs-r11.1-135+" {
        linux   /boot/vmlinuz-5.18.3-lfs-r11.1-135+ 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 r11.1-135+ > /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="r11.1-135+"
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="r11.1-135+"
ID=lfs
PRETTY_NAME="Linux From Scratch r11.1-135+"
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 *https://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 *make-ca* vous permettra de créer un ancrage de certificats de confiance, ce qui permet au système de vérifier les certificats SSL fournit par les serveurs distants (par exemple un site web en HTTPS).
- 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.
- Installing *sudo* may be useful for building packages as a non-`root` user and easily installing the resulting packages in your new system.
- 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*.
- Pour vous connecter à un point d'accès sans fil, installez *wpa_supplicant*.
- 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
  - /etc/sysconfig/ifconfig.eth0

Après cet intermède, démarrons notre toute nouvelle installation LFS pour la première fois ! Tout d'abord, quittez l'environnement chroot :

```
logout
```

Puis, démontez les systèmes de fichiers virtuels :

```
umount -v $LFS/dev/pts
umount -v $LFS/dev
umount -v $LFS/run
umount -v $LFS/proc
umount -v $LFS/sys
```

Si plusieurs partitions ont été créées, démontez les autres partitions avant de démonter la principale, comme ceci :

```
umount -v $LFS/usr
umount -v $LFS/home
umount -v $LFS
```

Démontez le système de fichiers de LFS :

```
umount -v $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 r11.1-135+*.

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-svn/*.

- 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 *File Transfer Protocol* ou 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 (FAI)

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 *Native Language Support* ou prise en charge 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> – Rédacteur de BLFS et meneur de ALFS
- *DJ Lucas* <dj@linuxfromscratch.org> – Rédacteur de LFS et de BLFS
- *Ken Moffat* <ken@linuxfromscratch.org> – Rédacteur 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 de 2008 à 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

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.ru

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> – Rédacteur du livre LFS
- Archaic <archaic@linuxfromscratch.org> – Rédacteur technique LFS, 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> – Rédacteur de LFS et BLFS
- *Alexander E. Patrakov* <alexander@linuxfromscratch.org> – Rédacteur Technique LFS, éditeur des traductions LFS, mainteneur du LiveCD LFS
- Simon Perreault
- *Scot Mc Pherson* <scot@linuxfromscratch.org> – Mainteneur de LFS NNTP Gateway
- *Douglas R. Reno* <renodr@linuxfromscratch.org> – Rédacteur 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> – Rédacteur 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 bonne compilation et la bonne installation de chaque paquet compilé dans LFS dépend d'un ou de 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, et parfois jusqu'à cinq 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 liste les paquets qui doivent être disponibles lorsqu'un programme ou une bibliothèque du paquet est utilisé à l'exécution. La troisième concerne les paquets qui, en plus de ceux de la première liste, doivent être disponibles afin de lancer les suites de test. La quatrième 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 vous être utiles. 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

Required at runtime: Attr et Glibc

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, Glibc, Grep, M4, Make, Perl, Sed et Texinfo

Required at runtime: Glibc

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

Required at runtime: Bash, Coreutils, Grep, M4, Make, 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
Required at runtime: Bash, Coreutils, Grep, M4, 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
Required at runtime: Glibc, Ncurses et Readline
La suite de tests dépend de: Expect et Shadow
Doit être installé avant: Aucun
Dépendances facultatives: *Xorg*

Bc

L'installation dépend de: Bash, Binutils, Coreutils, GCC, Glibc, Grep et Make
Required at runtime: Glibc, Ncurses et Readline
La suite de tests dépend de: Gawk
Doit être installé avant: 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
Required at runtime: Glibc et Zlib
La suite de tests dépend de: DejaGNU et Expect
Doit être installé avant: Aucun
Dépendances facultatives: *Elfutils*

Bison

L'installation dépend de: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, M4, Make, Perl et Sed
Required at runtime: Glibc
La suite de tests dépend de: Diffutils, Findutils et Flex
Doit être installé avant: Kbd et Tar
Dépendances facultatives: *Doxygen*

Bzip2

L'installation dépend de: Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Make, et Patch
Required at runtime: Glibc
La suite de tests dépend de: Aucun
Doit être installé avant: File
Dépendances facultatives: Aucun

Check

L'installation dépend de: Gawk, GCC, Grep, Make, Sed et Texinfo
Required at runtime: Bash et Gawk
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, OpenSSL, Patch, Perl, Sed et Texinfo
Required at runtime: Glibc
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: *Expect.pm* et *IO::Tty*

DejaGNU

L'installation dépend de: Bash, Coreutils, Diffutils, Expect, GCC, Grep, Make, Sed et Texinfo
Required at runtime: Expect et Bash
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
Required at runtime: Glibc
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

Required at runtime: Glibc 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

Required at runtime: Glibc, Kmod, Xz, Util-linux et Zlib.

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

Required at runtime: Glibc

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

Required at runtime: Glibc 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

Required at runtime: Glibc, Bzip2, Xz et Zlib

La suite de tests dépend de: Aucun

Doit être installé avant: Aucun

Dépendances facultatives: *libseccomp*

Findutils

L'installation dépend de: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, Make, Sed et Texinfo
Required at runtime: Bash et Glibc
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
Required at runtime: Bash, Glibc et M4
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
Required at runtime: Bash, Glibc et Mpfr
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
Required at runtime: Bash, Binutils, Glibc, Mpc et Python
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
Required at runtime: Bash, Glibc et Readline
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
Required at runtime: Acl, Bash, Gcc et Glibc
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
Required at runtime: Aucun
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
Required at runtime: GCC et Glibc
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
Required at runtime: GCC et Glibc
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
Required at runtime: Glibc
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
Required at runtime: GCC, Glibc et Perl
La suite de tests dépend de: Aucune suite de tests disponible
Doit être installé avant: Man-DB et Perl
Dépendances facultatives: *ghostscript* et *Uchardet*

GRUB

L'installation dépend de: Bash, Binutils, Bison, Coreutils, Diffutils, GCC, Gettext, Glibc, Grep, Make, Ncurses, Sed, Texinfo et Xz
Required at runtime: Bash, GCC, Gettext, Glibc, Xz et Sed.
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
Required at runtime: Bash et Glibc
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
Required at runtime: Aucun
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
Required at runtime: GCC, Glibc, Ncurses et Readline
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
Required at runtime: Autoconf, Automake, Bash, Glibc, Grep, Perl et Sed
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, Linux API Headers et Zlib
Required at runtime: Bash, Coreutils, Glibc, Libcap, Libelf et Zlib
La suite de tests dépend de: Aucune suite de tests disponible
Doit être installé avant: Aucun
Dépendances facultatives: *Berkeley DB* et *iptables*

Jinja2

L'installation dépend de: MarkupSafe et Python
Required at runtime: MarkupSafe et Python
La suite de tests dépend de: Aucune suite de tests disponible
Doit être installé avant: Systemd
Dépendances facultatives: Aucun

Kbd

L'installation dépend de: Bash, Binutils, Bison, Check, Coreutils, Flex, GCC, Gettext, Glibc, Gzip, Make, Patch et Sed
Required at runtime: Bash, Coreutils et Glibc
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, OpenSSL, Pkg-config, Sed, Xz et Zlib
Required at runtime: Glibc, Xz 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
Required at runtime: Glibc et Ncurses
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
Required at runtime: Glibc
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
Required at runtime: Glibc et Zlib
La suite de tests dépend de: Aucun
Doit être installé avant: IProute2 et Linux
Dépendances facultatives: Aucun

Libffi

L'installation dépend de: Bash, Binutils, Coreutils, GCC, Glibc, Make et Sed
Required at runtime: Glibc
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
Required at runtime: Glibc
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
Required at runtime: Autoconf, Automake, Bash, Binutils, Coreutils, File, GCC, Glibc, Grep, Make et Sed
La suite de tests dépend de: Autoconf, Automake et Findutils
Doit être installé avant: Aucun
Dépendances facultatives: Aucun

Linux

L'installation dépend de: Bash, Bc, Binutils, Coreutils, Diffutils, Findutils, GCC, Glibc, Grep, Gzip, Kmod, Libelf, Make, Ncurses, OpenSSL, Perl et Sed
Required at runtime: Aucun
La suite de tests dépend de: Aucune suite de tests disponible
Doit être installé avant: Aucun
Dépendances facultatives: *cpio*

Linux API Headers

L'installation dépend de: Bash, Binutils, Coreutils, Findutils, GCC, Glibc, Grep, Gzip, Make, Perl et Sed
Required at runtime: Aucun
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
Required at runtime: Bash et Glibc
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
Required at runtime: Glibc
La suite de tests dépend de: Perl et Procps-ng
Doit être installé avant: Aucun
Dépendances facultatives: *Guile*

Man-DB

L'installation dépend de: Bash, Binutils, Bzip2, Coreutils, Flex, GCC, GDBM, Gettext, Glibc, Grep, Groff, Gzip, Less, Libpipeline, Make, Sed et Xz

Required at runtime: Bash, GDBM, Groff, Glibc, Gzip, Less, Libpipeline et Zlib

La suite de tests dépend de: Util-linux

Doit être installé avant: Aucun

Dépendances facultatives: *libseccomp*

Man-Pages

L'installation dépend de: Bash, Coreutils et Make

Required at runtime: Aucun

La suite de tests dépend de: Aucune suite de tests disponible

Doit être installé avant: Aucun

Dépendances facultatives: Aucun

MarkupSafe

L'installation dépend de: Python

Required at runtime: Python

La suite de tests dépend de: Aucune suite de tests disponible

Doit être installé avant: Jinja2

Dépendances facultatives: Aucun

Meson

L'installation dépend de: Ninja et Python

Required at runtime: 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

Required at runtime: Glibc, GMP et MPFR

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

Required at runtime: Glibc et GMP

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

Required at runtime: Glibc

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

Required at runtime: GCC et Glibc

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

Required at runtime: Glibc et Perl

La suite de tests dépend de: Aucun

Doit être installé avant: Coreutils, Kmod et Linux

Dépendances facultatives: Aucun

Patch

L'installation dépend de: Bash, Binutils, Coreutils, GCC, Glibc, Grep, Make et Sed

Required at runtime: Glibc et Patch

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
Required at runtime: GDBM et Glibc
La suite de tests dépend de: Iana-Etc, Less et Procps-ng
Doit être installé avant: Autoconf
Dépendances facultatives: *Berkeley DB*

Pkg-config

L'installation dépend de: Bash, Binutils, Coreutils, Gawk, GCC, Glibc, Grep, Make, Popt et Sed
Required at runtime: Glibc
La suite de tests dépend de: Aucun
Doit être installé avant: Kmod
Dépendances facultatives: *Glib2*

Procps-ng

L'installation dépend de: Bash, Binutils, Coreutils, GCC, Glibc, Make et Ncurses
Required at runtime: Glibc
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
Required at runtime: Glibc et Ncurses
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, OpenSSL, Sed et Util-linux
Required at runtime: Bzip2, Expat, Gdbm, Glibc, Libffi, Ncurses, OpenSSL et Zlib
La suite de tests dépend de: GDB et
Doit être installé avant: Ninja
Dépendances facultatives: *Berkeley DB*, *libnsl*, *SQLite* et *Tk*

Readline

L'installation dépend de: Bash, Binutils, Coreutils, Gawk, GCC, Glibc, Grep, Make, Ncurses, Patch, Sed et Texinfo
Required at runtime: Glibc et Ncurses
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
Required at runtime: Acl, Attr et Glibc
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
Required at runtime: Glibc
La suite de tests dépend de: Aucune suite de tests disponible
Doit être installé avant: Coreutils
Dépendances facultatives: *CrackLib* et *Linux-PAM*

Sysklogd

L'installation dépend de: Binutils, Coreutils, GCC, Glibc, Make et Patch
Required at runtime: Glibc
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, Jinja2, Libcap, Meson, Sed, Util-linux et Zstd
Required at runtime: Acl, Attr, Glibc, Libcap et Util-linux
La suite de tests dépend de: Aucun
Doit être installé avant: Aucun
Dépendances facultatives: *btrfs-progs*, *cURL*, *cryptsetup*, *docbook-xml*, *docbook-xsl-nons*, *elfutils*, *Git*, *gnu-efi*, *GnuTLS*, *iptables*, *kexec-tools*, *libfido2*, *libgcrypt*, *libidn2*, *Libmicrohttpd*, *libpwquality*, *libseccomp*, *libxkbcommon*, *libxslt*, *Linux-PAM*, *lxml*, *LZ4*, *make-ca*, *p11-kit*, *PCRE2*, *Polkit*, *qemu*, *qrencode*, *quota-tools*, *rsync*, *Sphinx*, *tpm2-tss*, *Valgrind* et *zsh*

Sysvinit

L'installation dépend de: Binutils, Coreutils, GCC, Glibc, Make et Sed
Required at runtime: Glibc
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
Required at runtime: Acl, Attr, Bzip2, Glibc, Gzip et Xz
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
Required at runtime: Glibc et Zlib
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
Required at runtime: Glibc et Ncurses
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, Libcap, Make, Ncurses, Sed et Zlib
Required at runtime: Glibc, Libcap, Ncurses, Readline et Zlib
La suite de tests dépend de: Aucun
Doit être installé avant: Aucun
Dépendances facultatives: *Linux-PAM* et *smartmontools*

Vim

L'installation dépend de: Acl, Attr, Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Grep, Make, Ncurses et Sed

Required at runtime: Acl, Attr, Glibc, Python, Ncurses et Tcl

La suite de tests dépend de: Aucun

Doit être installé avant: Aucun

Dépendances facultatives: *Xorg*, *GTK+2*, *LessTif*, *Ruby* et *GPM*

wheel

L'installation dépend de: Python

Required at runtime: Python

La suite de tests dépend de: Aucune suite de tests disponible

Doit être installé avant: Jinja2

Dépendances facultatives: Aucun

XML::Parser

L'installation dépend de: Bash, Binutils, Coreutils, Expat, GCC, Glibc, Make et Perl

Required at runtime: Expat, Glibc 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

Required at runtime: Glibc

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

Required at runtime: Glibc

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
Required at runtime:	Glibc
La suite de tests dépend de:	Aucun
Doit être installé avant:	GCC et Systemd
Dépendances facultatives:	*LZ4*

Annexe D. Scripts de démarrage et de sysconfig version-20220327

Les scripts présents dans cette annexe sont listés dans le répertoire où ils sont généralement stockés. L'ordre est le suivant : `/etc/rc.d/init.d`, `/etc/sysconfig`, `/etc/sysconfig/network-devices` et `/etc/sysconfig/network-devices/services`. Au sein de chaque section, les fichiers sont listés dans l'ordre où ils sont normalement appelés.

D.1. /etc/rc.d/init.d/rc

Le script `rc` est le premier script appelé par init : il initialise le processus de démarrage.

```
#!/bin/bash
#############################################################################
# Begin rc
#
# Description : Main Run Level Control Script
#
# Authors     : Gerard Beekmans  - gerard@linuxfromscratch.org
#             : DJ Lucas - dj@linuxfromscratch.org
# Updates     : Bruce Dubbs - bdubbs@linuxfromscratch.org
#             : Pierre Labastie - pierre@linuxfromscratch.org
#
# Version     : LFS 7.0
#
# Notes       : Updates March 24th, 2022: new semantics of S/K files
#               - Instead of testing that S scripts were K scripts in the
#                 previous runlevel, test that they were not S scripts
#               - Instead of testing that K scripts were S scripts in the
#                 previous runlevel, test that they were not K scripts
#               - S scripts in runlevel 0 or 6 are now run with
#                "script start" (was "script stop" previously).
#############################################################################

. /lib/lsb/init-functions

print_error_msg()
{
   log_failure_msg
   # $i is set when called
   MSG="FAILURE:\n\nYou should not be reading this error message.\n\n"
   MSG="${MSG}It means that an unforeseen error took place in\n"
   MSG="${MSG}${i},\n"
   MSG="${MSG}which exited with a return value of ${error_value}.\n"

   MSG="${MSG}If you're able to track this error down to a bug in one of\n"
   MSG="${MSG}the files provided by the ${DISTRO_MINI} book,\n"
   MSG="${MSG}please be so kind to inform us at ${DISTRO_CONTACT}.\n"
   log_failure_msg "${MSG}"

   log_info_msg "Press Enter to continue..."
   wait_for_user
}

check_script_status()
{
   # $i is set when called
   if [ ! -f ${i} ]; then
      log_warning_msg "${i} is not a valid symlink."
      SCRIPT_STAT="1"
   fi

   if [ ! -x ${i} ]; then
      log_warning_msg "${i} is not executable, skipping."
```

```
      SCRIPT_STAT="1"
   fi
}

run()
{
   if [ -z $interactive ]; then
      ${1} ${2}
      return $?
   fi

   while true; do
      read -p "Run ${1} ${2} (Yes/no/continue)? " -n 1 runit
      echo

      case ${runit} in
         c | C)
            interactive=""
            ${i} ${2}
            ret=${?}
            break;
            ;;

         n | N)
            return 0
            ;;

         y | Y)
            ${i} ${2}
            ret=${?}
            break
            ;;
      esac
   done

   return $ret
}

# Read any local settings/overrides
[ -r /etc/sysconfig/rc.site ] && source /etc/sysconfig/rc.site

DISTRO=${DISTRO:-"Linux From Scratch"}
DISTRO_CONTACT=${DISTRO_CONTACT:-"lfs-dev@lists.linuxfromscratch.org (Registration required)"}
DISTRO_MINI=${DISTRO_MINI:-"LFS"}
IPROMPT=${IPROMPT:-"no"}

# These 3 signals will not cause our script to exit
trap "" INT QUIT TSTP

[ "${1}" != "" ] && runlevel=${1}

if [ "${runlevel}" == "" ]; then
   echo "Usage: ${0} <runlevel>" >&2
   exit 1
fi

previous=${PREVLEVEL}
[ "${previous}" == "" ] && previous=N

if [ ! -d /etc/rc.d/rc${runlevel}.d ]; then
   log_info_msg "/etc/rc.d/rc${runlevel}.d does not exist.\n"
   exit 1
fi

if [ "$runlevel" == "6" -o "$runlevel" == "0" ]; then IPROMPT="no"; fi

# Note: In ${LOGLEVEL:-7}, it is ':' 'dash' '7', not minus 7
```

```
if [ "$runlevel" == "S" ]; then
   [ -r /etc/sysconfig/console ] && source /etc/sysconfig/console
   dmesg -n "${LOGLEVEL:-7}"
fi

if [ "${IPROMPT}" == "yes" -a "${runlevel}" == "S" ]; then
   # The total length of the distro welcome string, without escape codes
   wlen=${wlen:-$(echo "Welcome to ${DISTRO}" | wc -c )}
   welcome_message=${welcome_message:-"Welcome to ${INFO}${DISTRO}${NORMAL}"}

   # The total length of the interactive string, without escape codes
   ilen=${ilen:-$(echo "Press 'I' to enter interactive startup" | wc -c )}
   i_message=${i_message:-"Press '${FAILURE}I${NORMAL}' to enter interactive startup"}


   # dcol and icol are spaces before the message to center the message
   # on screen. itime is the amount of wait time for the user to press a key
   wcol=$(( ( ${COLUMNS} - ${wlen} ) / 2 ))
   icol=$(( ( ${COLUMNS} - ${ilen} ) / 2 ))
   itime=${itime:-"3"}

   echo -e "\n\n"
   echo -e "\\033[${wcol}G${welcome_message}"
   echo -e "\\033[${icol}G${i_message}${NORMAL}"
   echo ""
   read -t "${itime}" -n 1 interactive 2>&1 > /dev/null
fi

# Make lower case
[ "${interactive}" == "I" ] && interactive="i"
[ "${interactive}" != "i" ] && interactive=""

# Read the state file if it exists from runlevel S
[ -r /run/interactive ] && source /run/interactive

# Stop all services marked as K, except if marked as K in the previous
# runlevel: it is the responsibility of the script to not try to kill
# a non running service
if [ "${previous}" != "N" ]; then
   for i in $(ls -v /etc/rc.d/rc${runlevel}.d/K* 2> /dev/null)
   do
      check_script_status
      if [ "${SCRIPT_STAT}" == "1" ]; then
         SCRIPT_STAT="0"
         continue
      fi

      suffix=${i#/etc/rc.d/rc${runlevel}.d/K[0-9][0-9]}
      [ -e /etc/rc.d/rc${previous}.d/K[0-9][0-9]$suffix ] && continue

      run ${i} stop
      error_value=${?}

      if [ "${error_value}" != "0" ]; then print_error_msg; fi
   done
fi

if [ "${previous}" == "N" ]; then export IN_BOOT=1; fi

if [ "$runlevel" == "6" -a -n "${FASTBOOT}" ]; then
   touch /fastboot
fi


# Start all services marked as S in this runlevel, except if marked as
# S in the previous runlevel
# it is the responsabily of the script to not try to start an already running
```

```
# service
for i in $( ls -v /etc/rc.d/rc${runlevel}.d/S* 2> /dev/null)
do

   if [ "${previous}" != "N" ]; then
      suffix=${i#/etc/rc.d/rc${runlevel}.d/S[0-9][0-9]}
      [ -e /etc/rc.d/rc${previous}.d/S[0-9][0-9]$suffix ] && continue
   fi

   check_script_status
   if [ "${SCRIPT_STAT}" == "1" ]; then
      SCRIPT_STAT="0"
      continue
   fi

   run ${i} start

   error_value=${?}

   if [ "${error_value}" != "0" ]; then print_error_msg; fi
done

# Store interactive variable on switch from runlevel S and remove if not
if [ "${runlevel}" == "S" -a "${interactive}" == "i" ]; then
    echo "interactive=\"i\"" > /run/interactive
else
    rm -f /run/interactive 2> /dev/null
fi

# Copy the boot log on initial boot only
if [ "${previous}" == "N" -a  "${runlevel}" != "S" ]; then
   cat $BOOTLOG >> /var/log/boot.log

   # Mark the end of boot
   echo "--------" >> /var/log/boot.log

   # Remove the temporary file
   rm -f $BOOTLOG 2> /dev/null
fi

# End rc
```

D.2. /lib/lsb/init-functions

```
#!/bin/sh
########################################################################
#
# Begin /lib/lsb/init-funtions
#
# Description : Run Level Control Functions
#
# Authors     : Gerard Beekmans - gerard@linuxfromscratch.org
#             : DJ Lucas - dj@linuxfromscratch.org
# Update      : Bruce Dubbs - bdubbs@linuxfromscratch.org
#
# Version     : LFS 7.0
#
# Notes       : With code based on Matthias Benkmann's simpleinit-msb
#               http://winterdrache.de/linux/newboot/index.html
#
#               The file should be located in /lib/lsb
#
########################################################################

## Environmental setup
# Setup default values for environment
umask 022
```

```
export PATH="/bin:/usr/bin:/sbin:/usr/sbin"

## Set color commands, used via echo
# Please consult `man console_codes for more information
# under the "ECMA-48 Set Graphics Rendition" section
#
# Warning: when switching from a 8bit to a 9bit font,
# the linux console will reinterpret the bold (1;) to
# the top 256 glyphs of the 9bit font.  This does
# not affect framebuffer consoles

NORMAL="\\033[0;39m"         # Standard console grey
SUCCESS="\\033[1;32m"        # Success is green
WARNING="\\033[1;33m"        # Warnings are yellow
FAILURE="\\033[1;31m"        # Failures are red
INFO="\\033[1;36m"           # Information is light cyan
BRACKET="\\033[1;34m"        # Brackets are blue

# Use a colored prefix
BMPREFIX="      "
SUCCESS_PREFIX="${SUCCESS}  *  ${NORMAL} "
FAILURE_PREFIX="${FAILURE}*****${NORMAL} "
WARNING_PREFIX="${WARNING} *** ${NORMAL} "
SKIP_PREFIX="${INFO}  S   ${NORMAL}"

SUCCESS_SUFFIX="${BRACKET}[${SUCCESS}  OK  ${BRACKET}]${NORMAL}"
FAILURE_SUFFIX="${BRACKET}[${FAILURE} FAIL ${BRACKET}]${NORMAL}"
WARNING_SUFFIX="${BRACKET}[${WARNING} WARN ${BRACKET}]${NORMAL}"
SKIP_SUFFIX="${BRACKET}[${INFO} SKIP ${BRACKET}]${NORMAL}"

BOOTLOG=/run/bootlog
KILLDELAY=3
SCRIPT_STAT="0"

# Set any user specified environment variables e.g. HEADLESS
[ -r /etc/sysconfig/rc.site ]  && . /etc/sysconfig/rc.site

## Screen Dimensions
# Find current screen size
if [ -z "${COLUMNS}" ]; then
   COLUMNS=$(stty size)
   COLUMNS=${COLUMNS##* }
fi

# When using remote connections, such as a serial port, stty size returns 0
if [ "${COLUMNS}" = "0" ]; then
   COLUMNS=80
fi

## Measurements for positioning result messages
COL=$((${COLUMNS} - 8))
WCOL=$((${COL} - 2))

## Set Cursor Position Commands, used via echo
SET_COL="\\033[${COL}G"      # at the $COL char
SET_WCOL="\\033[${WCOL}G"    # at the $WCOL char
CURS_UP="\\033[1A\\033[0G"   # Up one line, at the 0'th char
CURS_ZERO="\\033[0G"

##############################################################################
# start_daemon()                                                             #
# Usage: start_daemon [-f] [-n nicelevel] [-p pidfile] pathname [args...]    #
#                                                                            #
# Purpose: This runs the specified program as a daemon                       #
#                                                                            #
# Inputs: -f: (force) run the program even if it is already running.         #
#         -n nicelevel: specify a nice level. See 'man nice(1)'.             #
```

```
#         -p pidfile: use the specified file to determine PIDs.                #
#         pathname: the complete path to the specified program                 #
#         args: additional arguments passed to the program (pathname)          #
#                                                                              #
# Return values (as defined by LSB exit codes):                               #
#       0 - program is running or service is OK                                #
#       1 - generic or unspecified error                                       #
#       2 - invalid or excessive argument(s)                                   #
#       5 - program is not installed                                           #
################################################################################
start_daemon()
{
    local force=""
    local nice="0"
    local pidfile=""
    local pidlist=""
    local retval=""

    # Process arguments
    while true
    do
        case "${1}" in

            -f)
                force="1"
                shift 1
                ;;

            -n)
                nice="${2}"
                shift 2
                ;;

            -p)
                pidfile="${2}"
                shift 2
                ;;

            -*)
                return 2
                ;;

            *)
                program="${1}"
                break
                ;;
        esac
    done

    # Check for a valid program
    if [ ! -e "${program}" ]; then return 5; fi

    # Execute
    if [ -z "${force}" ]; then
        if [ -z "${pidfile}" ]; then
            # Determine the pid by discovery
            pidlist=`pidofproc "${1}"`
            retval="${?}"
        else
            # The PID file contains the needed PIDs
            # Note that by LSB requirement, the path must be given to pidofproc,
            # however, it is not used by the current implementation or standard.
            pidlist=`pidofproc -p "${pidfile}" "${1}"`
            retval="${?}"
        fi

        # Return a value ONLY
```

```
        # It is the init script's (or distribution's functions) responsibilty
        # to log messages!
        case "${retval}" in

            0)
                # Program is already running correctly, this is a
                # successful start.
                return 0
                ;;

            1)
                # Program is not running, but an invalid pid file exists
                # remove the pid file and continue
                rm -f "${pidfile}"
                ;;

            3)
                # Program is not running and no pidfile exists
                # do nothing here, let start_deamon continue.
                ;;

            *)
                # Others as returned by status values shall not be interpreted
                # and returned as an unspecified error.
                return 1
                ;;
        esac
    fi

    # Do the start!
    nice -n "${nice}" "${@}"
}

########################################################################################
# killproc()                                                                           #
# Usage: killproc [-p pidfile] pathname [signal]                                       #
#                                                                                      #
# Purpose: Send control signals to running processes                                   #
#                                                                                      #
# Inputs: -p pidfile, uses the specified pidfile                                       #
#         pathname, pathname to the specified program                                  #
#         signal, send this signal to pathname                                         #
#                                                                                      #
# Return values (as defined by LSB exit codes):                                       #
#       0 - program (pathname) has stopped/is already stopped or a                     #
#           running program has been sent specified signal and stopped                 #
#           successfully                                                               #
#       1 - generic or unspecified error                                               #
#       2 - invalid or excessive argument(s)                                           #
#       5 - program is not installed                                                   #
#       7 - program is not running and a signal was supplied                           #
########################################################################################
killproc()
{
    local pidfile
    local program
    local prefix
    local progname
    local signal="-TERM"
    local fallback="-KILL"
    local nosig
    local pidlist
    local retval
    local pid
    local delay="30"
    local piddead
    local dtime
```

```
    # Process arguments
    while true; do
        case "${1}" in
            -p)
                pidfile="${2}"
                shift 2
                ;;

             *)
                 program="${1}"
                 if [ -n "${2}" ]; then
                     signal="${2}"
                     fallback=""
                 else
                     nosig=1
                 fi

                 # Error on additional arguments
                 if [ -n "${3}" ]; then
                     return 2
                 else
                     break
                 fi
                 ;;
        esac
    done

    # Check for a valid program
    if [ ! -e "${program}" ]; then return 5; fi

    # Check for a valid signal
    check_signal "${signal}"
    if [ "${?}" -ne "0" ]; then return 2; fi

    # Get a list of pids
    if [ -z "${pidfile}" ]; then
        # determine the pid by discovery
        pidlist=`pidofproc "${1}"`
        retval="${?}"
    else
        # The PID file contains the needed PIDs
        # Note that by LSB requirement, the path must be given to pidofproc,
        # however, it is not used by the current implementation or standard.
        pidlist=`pidofproc -p "${pidfile}" "${1}"`
        retval="${?}"
    fi

    # Return a value ONLY
    # It is the init script's (or distribution's functions) responsibilty
    # to log messages!
    case "${retval}" in

        0)
            # Program is running correctly
            # Do nothing here, let killproc continue.
            ;;

        1)
            # Program is not running, but an invalid pid file exists
            # Remove the pid file.

            progname=${program##*/}

            if [[ -e "/run/${progname}.pid" ]]; then
                pidfile="/run/${progname}.pid"
                rm -f "${pidfile}"
```

```
            fi

            # This is only a success if no signal was passed.
            if [ -n "${nosig}" ]; then
                return 0
            else
                return 7
            fi
            ;;

        3)
            # Program is not running and no pidfile exists
            # This is only a success if no signal was passed.
            if [ -n "${nosig}" ]; then
                return 0
            else
                return 7
            fi
            ;;

        *)
            # Others as returned by status values shall not be interpreted
            # and returned as an unspecified error.
            return 1
            ;;
    esac

    # Perform different actions for exit signals and control signals
    check_sig_type "${signal}"

    if [ "${?}" -eq "0" ]; then # Signal is used to terminate the program

        # Account for empty pidlist (pid file still exists and no
        # signal was given)
        if [ "${pidlist}" != "" ]; then

            # Kill the list of pids
            for pid in ${pidlist}; do

                kill -0 "${pid}" 2> /dev/null

                if [ "${?}" -ne "0" ]; then
                    # Process is dead, continue to next and assume all is well
                    continue
                else
                    kill "${signal}" "${pid}" 2> /dev/null

                    # Wait up to ${delay}/10 seconds to for "${pid}" to
                    # terminate in 10ths of a second

                    while [ "${delay}" -ne "0" ]; do
                        kill -0 "${pid}" 2> /dev/null || piddead="1"
                        if [ "${piddead}" = "1" ]; then break; fi
                        sleep 0.1
                        delay="$(( ${delay} - 1 ))"
                    done

                    # If a fallback is set, and program is still running, then
                    # use the fallback
                    if [ -n "${fallback}" -a "${piddead}" != "1" ]; then
                        kill "${fallback}" "${pid}" 2> /dev/null
                        sleep 1
                        # Check again, and fail if still running
                        kill -0 "${pid}" 2> /dev/null && return 1
                    fi
                fi
            done
```

```
        fi

        # Check for and remove stale PID files.
        if [ -z "${pidfile}" ]; then
            # Find the basename of $program
            prefix=`echo "${program}" | sed 's/[^/]*$//'`
            progname=`echo "${program}" | sed "s@${prefix}@@"`

            if [ -e "/run/${progname}.pid" ]; then
                rm -f "/run/${progname}.pid" 2> /dev/null
            fi
        else
            if [ -e "${pidfile}" ]; then rm -f "${pidfile}" 2> /dev/null; fi
        fi

    # For signals that do not expect a program to exit, simply
    # let kill do its job, and evaluate kill's return for value

    else # check_sig_type - signal is not used to terminate program
        for pid in ${pidlist}; do
            kill "${signal}" "${pid}"
            if [ "${?}" -ne "0" ]; then return 1; fi
        done
    fi
}

################################################################################
# pidofproc()                                                                  #
# Usage: pidofproc [-p pidfile] pathname                                       #
#                                                                              #
# Purpose: This function returns one or more pid(s) for a particular daemon    #
#                                                                              #
# Inputs: -p pidfile, use the specified pidfile instead of pidof               #
#         pathname, path to the specified program                              #
#                                                                              #
# Return values (as defined by LSB status codes):                              #
#       0 - Success (PIDs to stdout)                                           #
#       1 - Program is dead, PID file still exists (remaining PIDs output)     #
#       3 - Program is not running (no output)                                 #
################################################################################
pidofproc()
{
    local pidfile
    local program
    local prefix
    local progname
    local pidlist
    local lpids
    local exitstatus="0"

    # Process arguments
    while true; do
        case "${1}" in

            -p)
                pidfile="${2}"
                shift 2
                ;;

            *)
                program="${1}"
                if [ -n "${2}" ]; then
                    # Too many arguments
                    # Since this is status, return unknown
                    return 4
                else
                    break
```

```
                fi
                ;;
        esac
    done

    # If a PID file is not specified, try and find one.
    if [ -z "${pidfile}" ]; then
        # Get the program's basename
        prefix=`echo "${program}" | sed 's/[^/]*$//'`

        if [ -z "${prefix}" ]; then
           progname="${program}"
        else
           progname=`echo "${program}" | sed "s@${prefix}@@"`
        fi

        # If a PID file exists with that name, assume that is it.
        if [ -e "/run/${progname}.pid" ]; then
            pidfile="/run/${progname}.pid"
        fi
    fi

    # If a PID file is set and exists, use it.
    if [ -n "${pidfile}" -a -e "${pidfile}" ]; then

        # Use the value in the first line of the pidfile
        pidlist=`/bin/head -n1 "${pidfile}"`
        # This can optionally be written as 'sed 1q' to repalce 'head -n1'
        # should LFS move /bin/head to /usr/bin/head
    else
        # Use pidof
        pidlist=`pidof "${program}"`
    fi

    # Figure out if all listed PIDs are running.
    for pid in ${pidlist}; do
        kill -0 ${pid} 2> /dev/null

        if [ "${?}" -eq "0" ]; then
            lpids="${lpids}${pid} "
        else
            exitstatus="1"
        fi
    done

    if [ -z "${lpids}" -a ! -f "${pidfile}" ]; then
        return 3
    else
        echo "${lpids}"
        return "${exitstatus}"
    fi
}

########################################################################################
# statusproc()                                                                         #
# Usage: statusproc [-p pidfile] pathname                                              #
#                                                                                      #
# Purpose: This function prints the status of a particular daemon to stdout            #
#                                                                                      #
# Inputs: -p pidfile, use the specified pidfile instead of pidof                       #
#         pathname, path to the specified program                                      #
#                                                                                      #
# Return values:                                                                       #
#       0 - Status printed                                                             #
#       1 - Input error. The daemon to check was not specified.                        #
########################################################################################
statusproc()
```

```
{
   local pidfile
   local pidlist

   if [ "${#}" = "0" ]; then
      echo "Usage: statusproc [-p pidfle] {program}"
      exit 1
   fi

   # Process arguments
   while true; do
       case "${1}" in

           -p)
               pidfile="${2}"
               shift 2
               ;;

           *)
               if [ -n "${2}" ]; then
                   echo "Too many arguments"
                   return 1
               else
                   break
               fi
               ;;
       esac
   done

   if [ -n "${pidfile}" ]; then
      pidlist=`pidofproc -p "${pidfile}" $@`
   else
      pidlist=`pidofproc $@`
   fi

   # Trim trailing blanks
   pidlist=`echo "${pidlist}" | sed -r 's/ +$//'`

   base="${1##*/}"

   if [ -n "${pidlist}" ]; then
      /bin/echo -e "${INFO}${base} is running with Process" \
         "ID(s) ${pidlist}.${NORMAL}"
   else
      if [ -n "${base}" -a -e "/run/${base}.pid" ]; then
         /bin/echo -e "${WARNING}${1} is not running but" \
            "/run/${base}.pid exists.${NORMAL}"
      else
         if [ -n "${pidfile}" -a -e "${pidfile}" ]; then
            /bin/echo -e "${WARNING}${1} is not running" \
               "but ${pidfile} exists.${NORMAL}"
         else
            /bin/echo -e "${INFO}${1} is not running.${NORMAL}"
         fi
      fi
   fi
}

##############################################################################
# timespec()                                                                 #
#                                                                            #
# Purpose: An internal utility function to format a timestamp                #
#          a boot log file.  Sets the STAMP variable.                        #
#                                                                            #
# Return value: Not used                                                     #
##############################################################################
timespec()
```

```
{
   STAMP="$(echo `date +"%b %d %T %:z"` `hostname`) "
   return 0
}

########################################################################################
# log_success_msg()                                                                    #
# Usage: log_success_msg ["message"]                                                   #
#                                                                                      #
# Purpose: Print a successful status message to the screen and                         #
#          a boot log file.                                                            #
#                                                                                      #
# Inputs: $@ - Message                                                                 #
#                                                                                      #
# Return values: Not used                                                              #
########################################################################################
log_success_msg()
{
    /bin/echo -n -e "${BMPREFIX}${@}"
    /bin/echo -e "${CURS_ZERO}${SUCCESS_PREFIX}${SET_COL}${SUCCESS_SUFFIX}"

    # Strip non-printable characters from log file
    logmessage=`echo "${@}" | sed 's/\\\033[^a-zA-Z]*.//g'`

    timespec
    /bin/echo -e "${STAMP} ${logmessage} OK" >> ${BOOTLOG}

    return 0
}

log_success_msg2()
{
    /bin/echo -n -e "${BMPREFIX}${@}"
    /bin/echo -e "${CURS_ZERO}${SUCCESS_PREFIX}${SET_COL}${SUCCESS_SUFFIX}"

    echo " OK" >> ${BOOTLOG}

    return 0
}

########################################################################################
# log_failure_msg()                                                                    #
# Usage: log_failure_msg ["message"]                                                   #
#                                                                                      #
# Purpose: Print a failure status message to the screen and                            #
#          a boot log file.                                                            #
#                                                                                      #
# Inputs: $@ - Message                                                                 #
#                                                                                      #
# Return values: Not used                                                              #
########################################################################################
log_failure_msg()
{
    /bin/echo -n -e "${BMPREFIX}${@}"
    /bin/echo -e "${CURS_ZERO}${FAILURE_PREFIX}${SET_COL}${FAILURE_SUFFIX}"

    # Strip non-printable characters from log file

    timespec
    logmessage=`echo "${@}" | sed 's/\\\033[^a-zA-Z]*.//g'`
    /bin/echo -e "${STAMP} ${logmessage} FAIL" >> ${BOOTLOG}

    return 0
}

log_failure_msg2()
{
```

```
    /bin/echo -n -e "${BMPREFIX}${@}"
    /bin/echo -e "${CURS_ZERO}${FAILURE_PREFIX}${SET_COL}${FAILURE_SUFFIX}"

    echo "FAIL" >> ${BOOTLOG}

    return 0
}

########################################################################################
# log_warning_msg()                                                                    #
# Usage: log_warning_msg ["message"]                                                   #
#                                                                                      #
# Purpose: Print a warning status message to the screen and                            #
#          a boot log file.                                                            #
#                                                                                      #
# Return values: Not used                                                              #
########################################################################################
log_warning_msg()
{
    /bin/echo -n -e "${BMPREFIX}${@}"
    /bin/echo -e "${CURS_ZERO}${WARNING_PREFIX}${SET_COL}${WARNING_SUFFIX}"

    # Strip non-printable characters from log file
    logmessage=`echo "${@}" | sed 's/\\\033[^a-zA-Z]*.//g'`
    timespec
    /bin/echo -e "${STAMP} ${logmessage} WARN" >> ${BOOTLOG}

    return 0
}

log_skip_msg()
{
    /bin/echo -n -e "${BMPREFIX}${@}"
    /bin/echo -e "${CURS_ZERO}${SKIP_PREFIX}${SET_COL}${SKIP_SUFFIX}"

    # Strip non-printable characters from log file
    logmessage=`echo "${@}" | sed 's/\\\033[^a-zA-Z]*.//g'`
    /bin/echo "SKIP" >> ${BOOTLOG}

    return 0
}

########################################################################################
# log_info_msg()                                                                       #
# Usage: log_info_msg message                                                          #
#                                                                                      #
# Purpose: Print an information message to the screen and                              #
#          a boot log file.  Does not print a trailing newline character.              #
#                                                                                      #
# Return values: Not used                                                              #
########################################################################################
log_info_msg()
{
    /bin/echo -n -e "${BMPREFIX}${@}"

    # Strip non-printable characters from log file
    logmessage=`echo "${@}" | sed 's/\\\033[^a-zA-Z]*.//g'`
    timespec
    /bin/echo -n -e "${STAMP} ${logmessage}" >> ${BOOTLOG}

    return 0
}

log_info_msg2()
{
    /bin/echo -n -e "${@}"
```

```
    # Strip non-printable characters from log file
    logmessage=`echo "${@}" | sed 's/\\\033[^a-zA-Z]*.//g'`
    /bin/echo -n -e "${logmessage}" >> ${BOOTLOG}

    return 0
}

##############################################################################
# evaluate_retval()                                                          #
# Usage: Evaluate a return value and print success or failure as appropriate #
#                                                                            #
# Purpose: Convenience function to terminate an info message                 #
#                                                                            #
# Return values: Not used                                                    #
##############################################################################
evaluate_retval()
{
   local error_value="${?}"

   if [ ${error_value} = 0 ]; then
      log_success_msg2
   else
      log_failure_msg2
   fi
}

##############################################################################
# check_signal()                                                             #
# Usage: check_signal [ -{signal} ]                                          #
#                                                                            #
# Purpose: Check for a valid signal.  This is not defined by any LSB draft,  #
#          however, it is required to check the signals to determine if the  #
#          signals chosen are invalid arguments to the other functions.      #
#                                                                            #
# Inputs: Accepts a single string value in the form of -{signal}             #
#                                                                            #
# Return values:                                                             #
#       0 - Success (signal is valid                                         #
#       1 - Signal is not valid                                              #
##############################################################################
check_signal()
{
    local valsig

    # Add error handling for invalid signals
    valsig=" -ALRM -HUP -INT -KILL -PIPE -POLL -PROF -TERM -USR1 -USR2"
    valsig="${valsig} -VTALRM -STKFLT -PWR -WINCH -CHLD -URG -TSTP -TTIN"
    valsig="${valsig} -TTOU -STOP -CONT -ABRT -FPE -ILL -QUIT -SEGV -TRAP"
    valsig="${valsig} -SYS -EMT -BUS -XCPU -XFSZ -0 -1 -2 -3 -4 -5 -6 -8 -9"
    valsig="${valsig} -11 -13 -14 -15 "

    echo "${valsig}" | grep -- " ${1} " > /dev/null

    if [ "${?}" -eq "0" ]; then
        return 0
    else
        return 1
    fi
}

##############################################################################
# check_sig_type()                                                           #
# Usage: check_signal [ -{signal} | {signal} ]                               #
#                                                                            #
# Purpose: Check if signal is a program termination signal or a control signal #
#          This is not defined by any LSB draft, however, it is required to  #
#          check the signals to determine if they are intended to end a      #
```

```
#          program or simply to control it.                                   #
#                                                                             #
# Inputs: Accepts a single string value in the form or -{signal} or {signal}  #
#                                                                             #
# Return values:                                                              #
#       0 - Signal is used for program termination                            #
#       1 - Signal is used for program control                                #
###############################################################################
check_sig_type()
{
    local valsig

    # The list of termination signals (limited to generally used items)
    valsig=" -ALRM -INT -KILL -TERM -PWR -STOP -ABRT -QUIT -2 -3 -6 -9 -14 -15 "

    echo "${valsig}" | grep -- " ${1} " > /dev/null

    if [ "${?}" -eq "0" ]; then
        return 0
    else
        return 1
    fi
}

###############################################################################
# wait_for_user()                                                             #
#                                                                             #
# Purpose: Wait for the user to respond if not a headless system              #
#                                                                             #
###############################################################################
wait_for_user()
{
   # Wait for the user by default
   [ "${HEADLESS=0}" = "0" ] && read ENTER
   return 0
}

###############################################################################
# is_true()                                                                   #
#                                                                             #
# Purpose: Utility to test if a variable is true | yes | 1                    #
#                                                                             #
###############################################################################
is_true()
{
   [ "$1" = "1" ] || [ "$1" = "yes" ] || [ "$1" = "true" ] ||  [ "$1" = "y" ] ||
   [ "$1" = "t" ]
}

# End /lib/lsb/init-functions
```

D.3. /etc/rc.d/init.d/mountvirtfs

```
#!/bin/sh
###############################################################################
# Begin mountvirtfs
#
# Description : Ensure proc, sysfs, run, and dev are mounted
#
# Authors     : Gerard Beekmans - gerard@linuxfromscratch.org
#               DJ Lucas - dj@linuxfromscratch.org
# Update      : Bruce Dubbs - bdubbs@linuxfromscratch.org
#
# Version     : LFS 7.0
#
###############################################################################
```

```
### BEGIN INIT INFO
# Provides:            mountvirtfs
# Required-Start:      $first
# Should-Start:
# Required-Stop:
# Should-Stop:
# Default-Start:       S
# Default-Stop:
# Short-Description:   Mounts various special fs needed at start
# Description:         Mounts /sys and /proc virtual (kernel) filesystems.
#                      Mounts /run (tmpfs) and /dev (devtmpfs).
#                      This is done only if they are not already mounted.
#                      with the kernel config proposed in the book, dev
#                      should be automatically mounted by the kernel.
# X-LFS-Provided-By:   LFS
### END INIT INFO

. /lib/lsb/init-functions

case "${1}" in
   start)
      # Make sure /run is available before logging any messages
      if ! mountpoint /run >/dev/null; then
         mount /run || failed=1
      fi

      mkdir -p /run/lock /run/shm
      chmod 1777 /run/shm /run/lock

      log_info_msg "Mounting virtual file systems: ${INFO}/run"

      if ! mountpoint /proc >/dev/null; then
         log_info_msg2 " ${INFO}/proc"
         mount -o nosuid,noexec,nodev /proc || failed=1
      fi

      if ! mountpoint /sys >/dev/null; then
         log_info_msg2 " ${INFO}/sys"
         mount -o nosuid,noexec,nodev /sys || failed=1
      fi

      if ! mountpoint /dev >/dev/null; then
         log_info_msg2 " ${INFO}/dev"
         mount -o mode=0755,nosuid /dev  || failed=1
      fi

      ln -sfn /run/shm /dev/shm

      (exit ${failed})
      evaluate_retval
      exit $failed
      ;;

   *)
      echo "Usage: ${0} {start}"
      exit 1
      ;;
esac

# End mountvirtfs
```

D.4. /etc/rc.d/init.d/modules

```
#!/bin/sh
########################################################################
# Begin modules
#
```

```
# Description : Module auto-loading script
#
# Authors     : Zack Winkles
#               DJ Lucas - dj@linuxfromscratch.org
# Update      : Bruce Dubbs - bdubbs@linuxfromscratch.org
#
# Version     : LFS 7.0
#
########################################################################

### BEGIN INIT INFO
# Provides:            modules
# Required-Start:      mountvirtfs
# Should-Start:
# Required-Stop:
# Should-Stop:
# Default-Start:       S
# Default-Stop:
# Short-Description:   Loads required modules.
# Description:         Loads modules listed in /etc/sysconfig/modules.
# X-LFS-Provided-By:   LFS
### END INIT INFO

# Assure that the kernel has module support.
[ -e /proc/modules ] || exit 0

. /lib/lsb/init-functions

case "${1}" in
   start)
      # Exit if there's no modules file or there are no
      # valid entries
      [ -r /etc/sysconfig/modules ]             || exit 0
      egrep -qv '^($|#)' /etc/sysconfig/modules || exit 0

      log_info_msg "Loading modules:"

      # Only try to load modules if the user has actually given us
      # some modules to load.

      while read module args; do

         # Ignore comments and blank lines.
         case "$module" in
            ""|"#"*) continue ;;
         esac

         # Attempt to load the module, passing any arguments provided.
         modprobe ${module} ${args} >/dev/null

         # Print the module name if successful, otherwise take note.
         if [ $? -eq 0 ]; then
            log_info_msg2 " ${module}"
         else
            failedmod="${failedmod} ${module}"
         fi
      done < /etc/sysconfig/modules

      # Print a message about successfully loaded modules on the correct line.
      log_success_msg2

      # Print a failure message with a list of any modules that
      # may have failed to load.
      if [ -n "${failedmod}" ]; then
         log_failure_msg "Failed to load modules:${failedmod}"
         exit 1
      fi
```

```
      ;;

   *)
      echo "Usage: ${0} {start}"
      exit 1
      ;;
esac

exit 0

# End modules
```

D.5. /etc/rc.d/init.d/udev

```
#!/bin/sh
########################################################################
# Begin udev
#
# Description : Udev cold-plugging script
#
# Authors     : Zack Winkles, Alexander E. Patrakov
#               DJ Lucas - dj@linuxfromscratch.org
# Update      : Bruce Dubbs - bdubbs@linuxfromscratch.org
#
# Version     : LFS 7.0
#
########################################################################

### BEGIN INIT INFO
# Provides:            udev $time
# Required-Start:      localnet
# Should-Start:        modules
# Required-Stop:
# Should-Stop:
# Default-Start:       S
# Default-Stop:
# Short-Description:   Populates /dev with device nodes.
# Description:         Mounts a tempfs on /dev and starts the udevd daemon.
#                      Device nodes are created as defined by udev.
# X-LFS-Provided-By:   LFS
### END INIT INFO

. /lib/lsb/init-functions

case "${1}" in
   start)
      log_info_msg "Populating /dev with device nodes... "
      if ! grep -q '[[:space:]]sysfs' /proc/mounts; then
         log_failure_msg2
         msg="FAILURE:\n\nUnable to create "
         msg="${msg}devices without a SysFS filesystem\n\n"
         msg="${msg}After you press Enter, this system "
         msg="${msg}will be halted and powered off.\n\n"
         log_info_msg "$msg"
         log_info_msg "Press Enter to continue..."
         wait_for_user
         /etc/rc.d/init.d/halt stop
      fi

      # Start the udev daemon to continually watch for, and act on,
      # uevents
      /sbin/udevd --daemon

      # Now traverse /sys in order to "coldplug" devices that have
      # already been discovered
      /sbin/udevadm trigger --action=add    --type=subsystems
      /sbin/udevadm trigger --action=add    --type=devices
```

```
      /sbin/udevadm trigger --action=change --type=devices

      # Now wait for udevd to process the uevents we triggered
      if ! is_true "$OMIT_UDEV_SETTLE"; then
         /sbin/udevadm settle
      fi

      # If any LVM based partitions are on the system, ensure they
      # are activated so they can be used.
      if [ -x /sbin/vgchange ]; then /sbin/vgchange -a y >/dev/null; fi

      log_success_msg2
      ;;

   *)
      echo "Usage ${0} {start}"
      exit 1
      ;;
esac

exit 0

# End udev
```

D.6. /etc/rc.d/init.d/swap

```
#!/bin/sh
########################################################################
# Begin swap
#
# Description : Swap Control Script
#
# Authors     : Gerard Beekmans - gerard@linuxfromscratch.org
#               DJ Lucas - dj@linuxfromscratch.org
# Update      : Bruce Dubbs - bdubbs@linuxfromscratch.org
#
# Version     : LFS 7.0
#
########################################################################

### BEGIN INIT INFO
# Provides:            swap
# Required-Start:      udev
# Should-Start:        modules
# Required-Stop:       localnet
# Should-Stop:         $local_fs
# Default-Start:       S
# Default-Stop:        0 6
# Short-Description:   Mounts and unmounts swap partitions.
# Description:         Mounts and unmounts swap partitions defined in
#                      /etc/fstab.
# X-LFS-Provided-By:   LFS
### END INIT INFO

. /lib/lsb/init-functions

case "${1}" in
   start)
      log_info_msg "Activating all swap files/partitions..."
      swapon -a
      evaluate_retval
      ;;

   stop)
      log_info_msg "Deactivating all swap files/partitions..."
      swapoff -a
      evaluate_retval
```

```
      ;;

   restart)
      ${0} stop
      sleep 1
      ${0} start
      ;;

   status)
      log_success_msg "Retrieving swap status."
      swapon -s
      ;;

   *)
      echo "Usage: ${0} {start|stop|restart|status}"
      exit 1
      ;;
esac

exit 0

# End swap
```

D.7. /etc/rc.d/init.d/setclock

```
#!/bin/sh
########################################################################
# Begin setclock
#
# Description : Setting Linux Clock
#
# Authors     : Gerard Beekmans - gerard@linuxfromscratch.org
#               DJ Lucas - dj@linuxfromscratch.org
# Update      : Bruce Dubbs - bdubbs@linuxfromscratch.org
#
# Version     : LFS 7.0
#
########################################################################

### BEGIN INIT INFO
# Provides:
# Required-Start:
# Should-Start:        modules
# Required-Stop:
# Should-Stop:         $syslog
# Default-Start:       S
# Default-Stop:
# Short-Description:   Stores and restores time from the hardware clock
# Description:         On boot, system time is obtained from hwclock.  The
#                      hardware clock can also be set on shutdown.
# X-LFS-Provided-By:   LFS
### END INIT INFO

. /lib/lsb/init-functions

[ -r /etc/sysconfig/clock ] && . /etc/sysconfig/clock


case "${UTC}" in
   yes|true|1)
      CLOCKPARAMS="${CLOCKPARAMS} --utc"
      ;;

   no|false|0)
      CLOCKPARAMS="${CLOCKPARAMS} --localtime"
      ;;
```

```
esac

case ${1} in
   start)
      hwclock --hctosys ${CLOCKPARAMS} >/dev/null
      ;;

   stop)
      log_info_msg "Setting hardware clock..."
      hwclock --systohc ${CLOCKPARAMS} >/dev/null
      evaluate_retval
      ;;

   *)
      echo "Usage: ${0} {start|stop}"
      exit 1
      ;;

esac

exit 0
```

D.8. /etc/rc.d/init.d/checkfs

```
#!/bin/sh
########################################################################
# Begin checkfs
#
# Description : File System Check
#
# Authors     : Gerard Beekmans - gerard@linuxfromscratch.org
#               A. Luebke - luebke@users.sourceforge.net
#               DJ Lucas - dj@linuxfromscratch.org
# Update      : Bruce Dubbs - bdubbs@linuxfromscratch.org
#
# Version     : LFS 7.0
#
# Based on checkfs script from LFS-3.1 and earlier.
#
# From man fsck
# 0    - No errors
# 1    - File system errors corrected
# 2    - System should be rebooted
# 4    - File system errors left uncorrected
# 8    - Operational error
# 16   - Usage or syntax error
# 32   - Fsck canceled by user request
# 128  - Shared library error
#
########################################################################

### BEGIN INIT INFO
# Provides:            checkfs
# Required-Start:      udev swap
# Should-Start:
# Required-Stop:
# Should-Stop:
# Default-Start:       S
# Default-Stop:
# Short-Description:   Checks local filesystems before mounting.
# Description:         Checks local filesystmes before mounting.
# X-LFS-Provided-By:   LFS
### END INIT INFO

. /lib/lsb/init-functions

case "${1}" in
```

```
   start)
      if [ -f /fastboot ]; then
         msg="/fastboot found, will omit "
         msg="${msg} file system checks as requested.\n"
         log_info_msg "${msg}"
         exit 0
      fi

      log_info_msg "Mounting root file system in read-only mode... "
      mount -n -o remount,ro / >/dev/null

      if [ ${?} != 0 ]; then
         log_failure_msg2
         msg="\n\nCannot check root "
         msg="${msg}filesystem because it could not be mounted "
         msg="${msg}in read-only mode.\n\n"
         msg="${msg}After you press Enter, this system will be "
         msg="${msg}halted and powered off.\n\n"
         log_failure_msg "${msg}"

         log_info_msg "Press Enter to continue..."
         wait_for_user
         /etc/rc.d/init.d/halt stop
      else
         log_success_msg2
      fi

      if [ -f /forcefsck ]; then
         msg="/forcefsck found, forcing file"
         msg="${msg} system checks as requested."
         log_success_msg "$msg"
         options="-f"
      else
         options=""
      fi

      log_info_msg "Checking file systems..."
      # Note: -a option used to be -p; but this fails e.g. on fsck.minix
      if is_true "$VERBOSE_FSCK"; then
        fsck ${options} -a -A -C -T
      else
        fsck ${options} -a -A -C -T >/dev/null
      fi

      error_value=${?}

      if [ "${error_value}" = 0 ]; then
         log_success_msg2
      fi

      if [ "${error_value}" = 1 ]; then
         msg="\nWARNING:\n\nFile system errors "
         msg="${msg}were found and have been corrected.\n"
         msg="${msg}      You may want to double-check that "
         msg="${msg}everything was fixed properly."
         log_warning_msg "$msg"
      fi

      if [ "${error_value}" = 2 -o "${error_value}" = 3 ]; then
         msg="\nWARNING:\n\nFile system errors "
         msg="${msg}were found and have been been "
         msg="${msg}corrected, but the nature of the "
         msg="${msg}errors require this system to be rebooted.\n\n"
         msg="${msg}After you press enter, "
         msg="${msg}this system will be rebooted\n\n"
         log_failure_msg "$msg"
```

```
         log_info_msg "Press Enter to continue..."
         wait_for_user
         reboot -f
      fi

      if [ "${error_value}" -gt 3 -a "${error_value}" -lt 16 ]; then
         msg="\nFAILURE:\n\nFile system errors "
         msg="${msg}were encountered that could not be "
         msg="${msg}fixed automatically.\nThis system "
         msg="${msg}cannot continue to boot and will "
         msg="${msg}therefore be halted until those "
         msg="${msg}errors are fixed manually by a "
         msg="${msg}System Administrator.\n\n"
         msg="${msg}After you press Enter, this system will be "
         msg="${msg}halted and powered off.\n\n"
         log_failure_msg "$msg"

         log_info_msg "Press Enter to continue..."
         wait_for_user
         /etc/rc.d/init.d/halt stop
      fi

      if [ "${error_value}" -ge 16 ]; then
         msg="FAILURE:\n\nUnexpected failure "
         msg="${msg}running fsck.  Exited with error "
         msg="${msg} code: ${error_value}.\n"
         log_info_msg $msg
         exit ${error_value}
      fi

      exit 0
      ;;
   *)
      echo "Usage: ${0} {start}"
      exit 1
      ;;
esac

# End checkfs
```

D.9. /etc/rc.d/init.d/mountfs

```
#!/bin/sh
########################################################################
# Begin mountfs
#
# Description : File System Mount Script
#
# Authors     : Gerard Beekmans - gerard@linuxfromscratch.org
#               DJ Lucas - dj@linuxfromscratch.org
# Update      : Bruce Dubbs - bdubbs@linuxfromscratch.org
#
# Version     : LFS 7.0
#
########################################################################

### BEGIN INIT INFO
# Provides:            $local_fs
# Required-Start:      udev checkfs
# Should-Start:        modules
# Required-Stop:       localnet
# Should-Stop:
# Default-Start:       S
# Default-Stop:        0 6
# Short-Description:   Mounts/unmounts local filesystems defined in /etc/fstab.
# Description:         Remounts root filesystem read/write and mounts all
#                      remaining local filesystems defined in /etc/fstab on
```

```
#                       start.  Remounts root filesystem read-only and unmounts
#                       remaining filesystems on stop.
# X-LFS-Provided-By:    LFS
### END INIT INFO

. /lib/lsb/init-functions

case "${1}" in
   start)
      log_info_msg "Remounting root file system in read-write mode..."
      mount --options remount,rw / >/dev/null
      evaluate_retval

      # Remove fsck-related file system watermarks.
      rm -f /fastboot /forcefsck

      # Make sure /dev/pts exists
      mkdir -p /dev/pts

      # This will mount all filesystems that do not have _netdev in
      # their option list.  _netdev denotes a network filesystem.

      log_info_msg "Mounting remaining file systems..."
      failed=0
      mount --all --test-opts no_netdev >/dev/null || failed=1
      evaluate_retval
      exit $failed
      ;;

   stop)
      # Don't unmount virtual file systems like /run
      log_info_msg "Unmounting all other currently mounted file systems..."
      # Ensure any loop devies are removed
      losetup -D
      umount --all --detach-loop --read-only \
             --types notmpfs,nosysfs,nodevtmpfs,noproc,nodevpts >/dev/null
      evaluate_retval

      # Make sure / is mounted read only (umount bug)
      mount --options remount,ro /

      # Make all LVM volume groups unavailable, if appropriate
      # This fails if swap or / are on an LVM partition
      #if [ -x /sbin/vgchange ]; then /sbin/vgchange -an > /dev/null; fi
      ;;

   *)
      echo "Usage: ${0} {start|stop}"
      exit 1
      ;;
esac

# End mountfs
```

D.10. /etc/rc.d/init.d/udev_retry

```
#!/bin/sh
########################################################################
# Begin udev_retry
#
# Description : Udev cold-plugging script (retry)
#
# Authors     : Alexander E. Patrakov
#               DJ Lucas - dj@linuxfromscratch.org
# Update      : Bruce Dubbs - bdubbs@linuxfromscratch.org
#               Bryan Kadzban -
#
```

```
# Version     : LFS 7.0
#
########################################################################

### BEGIN INIT INFO
# Provides:            udev_retry
# Required-Start:      udev
# Should-Start:        $local_fs cleanfs
# Required-Stop:
# Should-Stop:
# Default-Start:       S
# Default-Stop:
# Short-Description:   Replays failed uevents and creates additional devices.
# Description:         Replays any failed uevents that were skipped due to
#                      slow hardware initialization, and creates those needed
#                      device nodes
# X-LFS-Provided-By:   LFS
### END INIT INFO

. /lib/lsb/init-functions

case "${1}" in
   start)
      log_info_msg "Retrying failed uevents, if any..."

      # As of udev-186, the --run option is no longer valid
      #rundir=$(/sbin/udevadm info --run)
      rundir=/run/udev
      # From Debian: "copy the rules generated before / was mounted
      # read-write":

      for file in ${rundir}/tmp-rules--*; do
         dest=${file##*tmp-rules--}
         [ "$dest" = '*' ] && break
         cat $file >> /etc/udev/rules.d/$dest
         rm -f $file
      done

      # Re-trigger the uevents that may have failed,
      # in hope they will succeed now
      /bin/sed -e 's/#.*$//' /etc/sysconfig/udev_retry | /bin/grep -v '^$' | \
      while read line ; do
         for subsystem in $line ; do
            /sbin/udevadm trigger --subsystem-match=$subsystem --action=add
         done
      done

      # Now wait for udevd to process the uevents we triggered
      if ! is_true "$OMIT_UDEV_RETRY_SETTLE"; then
         /sbin/udevadm settle
      fi

      evaluate_retval
      ;;

   *)
      echo "Usage ${0} {start}"
      exit 1
      ;;
esac

exit 0

# End udev_retry
```

D.11. /etc/rc.d/init.d/cleanfs

```
#!/bin/sh
########################################################################
# Begin cleanfs
#
# Description : Clean file system
#
# Authors     : Gerard Beekmans - gerard@linuxfromscratch.org
#               DJ Lucas - dj@linuxfromscratch.org
# Update      : Bruce Dubbs - bdubbs@linuxfromscratch.org
#
# Version     : LFS 7.0
#
########################################################################

### BEGIN INIT INFO
# Provides:            cleanfs
# Required-Start:      $local_fs
# Should-Start:
# Required-Stop:
# Should-Stop:
# Default-Start:       S
# Default-Stop:
# Short-Description:   Cleans temporary directories early in the boot process.
# Description:         Cleans temporary directories /run, /var/lock, and
#                      optionally, /tmp.  cleanfs also creates /run/utmp
#                      and any files defined in /etc/sysconfig/createfiles.
# X-LFS-Provided-By:   LFS
### END INIT INFO

. /lib/lsb/init-functions

# Function to create files/directory on boot.
create_files()
{
   # Input to file descriptor 9 and output to stdin (redirection)
   exec 9>&0 < /etc/sysconfig/createfiles

   while read name type perm usr grp dtype maj min junk
   do
      # Ignore comments and blank lines.
      case "${name}" in
         ""|\#*) continue ;;
      esac

      # Ignore existing files.
      if [ ! -e "${name}" ]; then
         # Create stuff based on its type.
         case "${type}" in
            dir)
               mkdir "${name}"
               ;;
            file)
               :> "${name}"
               ;;
            dev)
               case "${dtype}" in
                  char)
                     mknod "${name}" c ${maj} ${min}
                     ;;
                  block)
                     mknod "${name}" b ${maj} ${min}
                     ;;
                  pipe)
                     mknod "${name}" p
                     ;;
```

```
                  *)
                     log_warning_msg "\nUnknown device type: ${dtype}"
                     ;;
               esac
               ;;
            *)
               log_warning_msg "\nUnknown type: ${type}"
               continue
               ;;
         esac

         # Set up the permissions, too.
         chown ${usr}:${grp} "${name}"
         chmod ${perm} "${name}"
      fi
   done

   # Close file descriptor 9 (end redirection)
   exec 0>&9 9>&-
   return 0
}

case "${1}" in
   start)
      log_info_msg "Cleaning file systems:"

      if [ "${SKIPTMPCLEAN}" = "" ]; then
         log_info_msg2 " /tmp"
         cd /tmp &&
         find . -xdev -mindepth 1 ! -name lost+found -delete || failed=1
      fi

      > /run/utmp

      if grep -q '^utmp:' /etc/group ; then
         chmod 664 /run/utmp
         chgrp utmp /run/utmp
      fi

      (exit ${failed})
      evaluate_retval

      if egrep -qv '^(#|$)' /etc/sysconfig/createfiles 2>/dev/null; then
         log_info_msg "Creating files and directories... "
         create_files      # Always returns 0
         evaluate_retval
      fi

      exit $failed
      ;;
   *)
      echo "Usage: ${0} {start}"
      exit 1
      ;;
esac

# End cleanfs
```

D.12. /etc/rc.d/init.d/console

```
#!/bin/sh
########################################################################
# Begin console
#
# Description : Sets keymap and screen font
#
# Authors     : Gerard Beekmans - gerard@linuxfromscratch.org
```

```
#               Alexander E. Patrakov
#               DJ Lucas - dj@linuxfromscratch.org
# Update      : Bruce Dubbs - bdubbs@linuxfromscratch.org
#
# Version     : LFS 7.0
#
########################################################################

### BEGIN INIT INFO
# Provides:            console
# Required-Start:      $local_fs
# Should-Start:        udev_retry
# Required-Stop:
# Should-Stop:
# Default-Start:       S
# Default-Stop:
# Short-Description:   Sets up a localised console.
# Description:         Sets up fonts and language settings for the user's
#                      local as defined by /etc/sysconfig/console.
# X-LFS-Provided-By:   LFS
### END INIT INFO

. /lib/lsb/init-functions

# Native English speakers probably don't have /etc/sysconfig/console at all
[ -r /etc/sysconfig/console ] && . /etc/sysconfig/console

failed=0

case "${1}" in
   start)
      # See if we need to do anything
      if [ -z "${KEYMAP}"         ] && [ -z "${KEYMAP_CORRECTIONS}" ] &&
         [ -z "${FONT}"           ] && [ -z "${LEGACY_CHARSET}"     ] &&
         ! is_true "${UNICODE}"; then
         exit 0
      fi

      # There should be no bogus failures below this line!
      log_info_msg "Setting up Linux console..."

      # Figure out if a framebuffer console is used
      [ -d /sys/class/graphics/fb0 ] && use_fb=1 || use_fb=0

      # Figure out the command to set the console into the
      # desired mode
      is_true "${UNICODE}" &&
         MODE_COMMAND="echo -en '\033%G' && kbd_mode -u" ||
         MODE_COMMAND="echo -en '\033%@\033(K' && kbd_mode -a"

      # On framebuffer consoles, font has to be set for each vt in
      # UTF-8 mode. This doesn't hurt in non-UTF-8 mode also.

      ! is_true "${use_fb}" || [ -z "${FONT}" ] ||
         MODE_COMMAND="${MODE_COMMAND} && setfont ${FONT}"

      # Apply that command to all consoles mentioned in
      # /etc/inittab. Important: in the UTF-8 mode this should
      # happen before setfont, otherwise a kernel bug will
      # show up and the unicode map of the font will not be
      # used.

      for TTY in `grep '^[^#].*respawn:/sbin/agetty' /etc/inittab |
         grep -o '\btty[[:digit:]]*\b'`
      do
         openvt -f -w -c ${TTY#tty} -- \
            /bin/sh -c "${MODE_COMMAND}" || failed=1
```

```
      done

      # Set the font (if not already set above) and the keymap
      [ "${use_fb}" == "1" ] || [ -z "${FONT}" ] || setfont $FONT || failed=1

      [ -z "${KEYMAP}" ] ||
         loadkeys ${KEYMAP} >/dev/null 2>&1 ||
         failed=1

      [ -z "${KEYMAP_CORRECTIONS}" ] ||
         loadkeys ${KEYMAP_CORRECTIONS} >/dev/null 2>&1 ||
         failed=1

      # Convert the keymap from $LEGACY_CHARSET to UTF-8
      [ -z "$LEGACY_CHARSET" ] ||
         dumpkeys -c "$LEGACY_CHARSET" | loadkeys -u >/dev/null 2>&1 ||
         failed=1

      # If any of the commands above failed, the trap at the
      # top would set $failed to 1
      ( exit $failed )
      evaluate_retval

      exit $failed
      ;;

   *)
      echo "Usage:  ${0} {start}"
      exit 1
      ;;
esac

# End console
```

D.13. /etc/rc.d/init.d/localnet

```
#!/bin/sh
########################################################################
# Begin localnet
#
# Description : Loopback device
#
# Authors     : Gerard Beekmans  - gerard@linuxfromscratch.org
#               DJ Lucas - dj@linuxfromscratch.org
# Update      : Bruce Dubbs - bdubbs@linuxfromscratch.org
#
# Version     : LFS 7.0
#
########################################################################

### BEGIN INIT INFO
# Provides:            localnet
# Required-Start:      mountvirtfs
# Should-Start:        modules
# Required-Stop:
# Should-Stop:
# Default-Start:       S
# Default-Stop:        0 6
# Short-Description:   Starts the local network.
# Description:         Sets the hostname of the machine and starts the
#                      loopback interface.
# X-LFS-Provided-By:   LFS
### END INIT INFO

. /lib/lsb/init-functions
[ -r /etc/sysconfig/network ] && . /etc/sysconfig/network
[ -r /etc/hostname ] && HOSTNAME=`cat /etc/hostname`
```

```
case "${1}" in
   start)
      log_info_msg "Bringing up the loopback interface..."
      ip addr add 127.0.0.1/8 label lo dev lo
      ip link set lo up
      evaluate_retval

      log_info_msg "Setting hostname to ${HOSTNAME}..."
      hostname ${HOSTNAME}
      evaluate_retval
      ;;

   stop)
      log_info_msg "Bringing down the loopback interface..."
      ip link set lo down
      evaluate_retval
      ;;

   restart)
      ${0} stop
      sleep 1
      ${0} start
      ;;

   status)
      echo "Hostname is: $(hostname)"
      ip link show lo
      ;;

   *)
      echo "Usage: ${0} {start|stop|restart|status}"
      exit 1
      ;;
esac

exit 0

# End localnet
```

D.14. /etc/rc.d/init.d/sysctl

```
#!/bin/sh
########################################################################
# Begin sysctl
#
# Description : File uses /etc/sysctl.conf to set kernel runtime
#               parameters
#
# Authors     : Nathan Coulson (nathan@linuxfromscratch.org)
#               Matthew Burgress (matthew@linuxfromscratch.org)
#               DJ Lucas - dj@linuxfromscratch.org
# Update      : Bruce Dubbs - bdubbs@linuxfromscratch.org
#
# Version     : LFS 7.0
#
########################################################################

### BEGIN INIT INFO
# Provides:            sysctl
# Required-Start:      mountvirtfs
# Should-Start:        console
# Required-Stop:
# Should-Stop:
# Default-Start:       S
# Default-Stop:
# Short-Description:   Makes changes to the proc filesystem
```

```
# Description:         Makes changes to the proc filesystem as defined in
#                      /etc/sysctl.conf.  See 'man sysctl(8)'.
# X-LFS-Provided-By:   LFS
### END INIT INFO

. /lib/lsb/init-functions

case "${1}" in
   start)
      if [ -f "/etc/sysctl.conf" ]; then
         log_info_msg "Setting kernel runtime parameters..."
         sysctl -q -p
         evaluate_retval
      fi
      ;;

   status)
      sysctl -a
      ;;

   *)
      echo "Usage: ${0} {start|status}"
      exit 1
      ;;
esac

exit 0

# End sysctl
```

D.15. /etc/rc.d/init.d/sysklogd

```
#!/bin/sh
########################################################################
# Begin sysklogd
#
# Description : Sysklogd loader
#
# Authors     : Gerard Beekmans - gerard@linuxfromscratch.org
#               DJ Lucas - dj@linuxfromscratch.org
# Update      : Bruce Dubbs - bdubbs@linuxfromscratch.org
#
# Version     : LFS 7.0
#
########################################################################

### BEGIN INIT INFO
# Provides:            $syslog
# Required-Start:      $first localnet
# Should-Start:
# Required-Stop:       $local_fs
# Should-Stop:         sendsignals
# Default-Start:       2 3 4 5
# Default-Stop:        0 1 6
# Short-Description:   Starts kernel and system log daemons.
# Description:         Starts kernel and system log daemons.
#                      /etc/fstab.
# X-LFS-Provided-By:   LFS
### END INIT INFO

. /lib/lsb/init-functions

case "${1}" in
   start)
      log_info_msg "Starting system log daemon..."
      parms=${SYSKLOGD_PARMS-'-m 0'}
      start_daemon /sbin/syslogd $parms
```

```
      evaluate_retval

      log_info_msg "Starting kernel log daemon..."
      start_daemon /sbin/klogd
      evaluate_retval
      ;;

   stop)
      log_info_msg "Stopping kernel log daemon..."
      killproc /sbin/klogd
      evaluate_retval

      log_info_msg "Stopping system log daemon..."
      killproc /sbin/syslogd
      evaluate_retval
      ;;

   reload)
      log_info_msg "Reloading system log daemon config file..."
      pid=`pidofproc syslogd`
      kill -HUP "${pid}"
      evaluate_retval
      ;;

   restart)
      ${0} stop
      sleep 1
      ${0} start
      ;;

   status)
      statusproc /sbin/syslogd
      statusproc klogd
      ;;

   *)
      echo "Usage: ${0} {start|stop|reload|restart|status}"
      exit 1
      ;;
esac

exit 0

# End sysklogd
```

D.16. /etc/rc.d/init.d/network

```
#!/bin/sh
########################################################################
# Begin network
#
# Description : Network Control Script
#
# Authors     : Gerard Beekmans - gerard@linuxfromscratch.org
#               Nathan Coulson - nathan@linuxfromscratch.org
#               Kevin P. Fleming - kpfleming@linuxfromscratch.org
#               DJ Lucas - dj@linuxfromscratch.org
# Update      : Bruce Dubbs - bdubbs@linuxfromscratch.org
#
# Version     : LFS 7.0
#
########################################################################

### BEGIN INIT INFO
# Provides:            $network
# Required-Start:      $local_fs localnet swap
# Should-Start:        $syslog firewalld iptables nftables
```

```
# Required-Stop:       $local_fs localnet swap
# Should-Stop:         $syslog firewalld iptables nftables
# Default-Start:       2 3 4 5
# Default-Stop:        0 1 6
# Short-Description:   Starts and configures network interfaces.
# Description:         Starts and configures network interfaces.
# X-LFS-Provided-By:   LFS
### END INIT INFO

case "${1}" in
   start)
      # if the default route exists, network is already configured
      if ip route | grep -q "^default"; then return 0; fi
      # Start all network interfaces
      for file in /etc/sysconfig/ifconfig.*
      do
         interface=${file##*/ifconfig.}

         # Skip if $file is * (because nothing was found)
         if [ "${interface}" = "*" ]; then continue; fi

         /sbin/ifup ${interface}
      done
      ;;

   stop)
      # Unmount any network mounted file systems
       umount --all --force --types nfs,cifs,nfs4

      # Reverse list
      net_files=""
      for file in  /etc/sysconfig/ifconfig.*
      do
         net_files="${file} ${net_files}"
      done

      # Stop all network interfaces
      for file in ${net_files}
      do
         interface=${file##*/ifconfig.}

         # Skip if $file is * (because nothing was found)
         if [ "${interface}" = "*" ]; then continue; fi

         # See if interface exists
         if [ ! -e /sys/class/net/$interface ]; then continue; fi

         # Is interface UP?
         ip link show $interface 2>/dev/null | grep -q "state UP"
         if [ $? -ne 0 ];  then continue; fi

         /sbin/ifdown ${interface}
      done
      ;;

   restart)
      ${0} stop
      sleep 1
      ${0} start
      ;;

   *)
      echo "Usage: ${0} {start|stop|restart}"
      exit 1
      ;;
esac
```

```
exit 0

# End network
```

D.17. /etc/rc.d/init.d/sendsignals

```
#!/bin/sh
########################################################################
# Begin sendsignals
#
# Description : Sendsignals Script
#
# Authors     : Gerard Beekmans - gerard@linuxfromscratch.org
#               DJ Lucas - dj@linuxfromscratch.org
# Update      : Bruce Dubbs - bdubbs@linuxfromscratch.org
#
# Version     : LFS 7.0
#
########################################################################

### BEGIN INIT INFO
# Provides:            sendsignals
# Required-Start:
# Should-Start:
# Required-Stop:       $local_fs swap localnet
# Should-Stop:
# Default-Start:
# Default-Stop:        0 6
# Short-Description:   Attempts to kill remaining processes.
# Description:         Attempts to kill remaining processes.
# X-LFS-Provided-By:   LFS
### END INIT INFO

. /lib/lsb/init-functions

case "${1}" in
   stop)
      omit=$(pidof mdmon)
      [ -n "$omit" ] && omit="-o $omit"

      log_info_msg "Sending all processes the TERM signal..."
      killall5 -15 $omit
      error_value=${?}

      sleep ${KILLDELAY}

      if [ "${error_value}" = 0 -o "${error_value}" = 2 ]; then
         log_success_msg
      else
         log_failure_msg
      fi

      log_info_msg "Sending all processes the KILL signal..."
      killall5 -9 $omit
      error_value=${?}

      sleep ${KILLDELAY}

      if [ "${error_value}" = 0 -o "${error_value}" = 2 ]; then
         log_success_msg
      else
         log_failure_msg
      fi
      ;;

   *)
      echo "Usage: ${0} {stop}"
```

```
      exit 1
      ;;

esac

exit 0

# End sendsignals
```

D.18. /etc/rc.d/init.d/reboot

```
#!/bin/sh
########################################################################
# Begin reboot
#
# Description : Reboot Scripts
#
# Authors     : Gerard Beekmans - gerard@linuxfromscratch.org
#               DJ Lucas - dj@linuxfromscratch.org
# Updates     : Bruce Dubbs - bdubbs@linuxfromscratch.org
#             : Pierre Labastie - pierre@linuxfromscratch.org
#
# Version     : LFS 7.0
#
# Notes       : Update March 24th, 2022: change "stop" to "start".
#               Add the $last facility to Required-start
#
########################################################################

### BEGIN INIT INFO
# Provides:            reboot
# Required-Start:      $last
# Should-Start:
# Required-Stop:
# Should-Stop:
# Default-Start:       6
# Default-Stop:
# Short-Description:   Reboots the system.
# Description:         Reboots the System.
# X-LFS-Provided-By:   LFS
### END INIT INFO

. /lib/lsb/init-functions

case "${1}" in
   start)
      log_info_msg "Restarting system..."
      reboot -d -f -i
      ;;

   *)
      echo "Usage: ${0} {start}"
      exit 1
      ;;

esac

# End reboot
```

D.19. /etc/rc.d/init.d/halt

```
#!/bin/sh
########################################################################
# Begin halt
#
```

```
# Description : Halt Script
#
# Authors     : Gerard Beekmans - gerard@linuxfromscratch.org
#               DJ Lucas - dj@linuxfromscratch.org
# Update      : Bruce Dubbs - bdubbs@linuxfromscratch.org
#             : Pierre Labastie - pierre@linuxfromscratch.org
#
# Version     : LFS 7.0
#
# Notes       : Update March 24th, 2022: change "stop" to "start".
#               Add the $last facility to Required-start
#
########################################################################

### BEGIN INIT INFO
# Provides:            halt
# Required-Start:      $last
# Should-Start:
# Required-Stop:
# Should-Stop:
# Default-Start:       0
# Default-Stop:
# Short-Description:   Halts the system.
# Description:         Halts the System.
# X-LFS-Provided-By:   LFS
### END INIT INFO

case "${1}" in
   start)
      halt -d -f -i -p
      ;;

   *)
      echo "Usage: {start}"
      exit 1
      ;;
esac

# End halt
```

D.20. /etc/rc.d/init.d/template

```
#!/bin/sh
########################################################################
# Begin scriptname
#
# Description :
#
# Authors     :
#
# Version     : LFS x.x
#
# Notes       :
#
########################################################################

### BEGIN INIT INFO
# Provides:            template
# Required-Start:
# Should-Start:
# Required-Stop:
# Should-Stop:
# Default-Start:
# Default-Stop:
# Short-Description:
# Description:
# X-LFS-Provided-By:
```

```
### END INIT INFO

. /lib/lsb/init-functions

case "${1}" in
   start)
      log_info_msg "Starting..."
    # if it is possible to use start_daemon
      start_daemon fully_qualified_path
    # if it is not possible to use start_daemon
    # (command to start the daemon is not simple enough)
      if ! pidofproc daemon_name_as_reported_by_ps >/dev/null; then
         command_to_start_the_service
      fi
      evaluate_retval
      ;;

   stop)
      log_info_msg "Stopping..."
    # if it is possible to use killproc
      killproc fully_qualified_path
    # if it is not possible to use killproc
    # (the daemon shoudn't be stopped by killing it)
      if pidofproc daemon_name_as_reported_by_ps >/dev/null; then
         command_to_stop_the_service
      fi
      evaluate_retval
      ;;

   restart)
      ${0} stop
      sleep 1
      ${0} start
      ;;

   *)
      echo "Usage: ${0} {start|stop|restart}"
      exit 1
      ;;
esac

exit 0

# End scriptname
```

D.21. /etc/sysconfig/modules

```
################################################################################
# Begin /etc/sysconfig/modules
#
# Description : Module auto-loading configuration
#
# Authors     :
#
# Version     : 00.00
#
# Notes       : The syntax of this file is as follows:
#          <module> [<arg1> <arg2> ...]
#
# Each module should be on its own line, and any options that you want
# passed to the module should follow it.  The line deliminator is either
# a space or a tab.
################################################################################

# End /etc/sysconfig/modules
```

D.22. /etc/sysconfig/createfiles

```
################################################################################
# Begin /etc/sysconfig/createfiles
#
# Description : Createfiles script config file
#
# Authors     :
#
# Version     : 00.00
#
# Notes       : The syntax of this file is as follows:
#         if type is equal to "file" or "dir"
#          <filename> <type> <permissions> <user> <group>
#         if type is equal to "dev"
#          <filename> <type> <permissions> <user> <group> <devtype>
#             <major> <minor>
#
#         <filename> is the name of the file which is to be created
#         <type> is either file, dir, or dev.
#               file creates a new file
#               dir creates a new directory
#               dev creates a new device
#         <devtype> is either block, char or pipe
#               block creates a block device
#               char creates a character deivce
#               pipe creates a pipe, this will ignore the <major> and
#           <minor> fields
#         <major> and <minor> are the major and minor numbers used for
#     the device.
################################################################################

# End /etc/sysconfig/createfiles
```

D.23. /etc/sysconfig/udev-retry

```
################################################################################
# Begin /etc/sysconfig/udev_retry
#
# Description : udev_retry script configuration
#
# Authors     :
#
# Version     : 00.00
#
# Notes       : Each subsystem that may need to be re-triggered after mountfs
#               runs should be listed in this file.  Probable subsystems to be
#               listed here are rtc (due to /var/lib/hwclock/adjtime) and sound
#               (due to both /var/lib/alsa/asound.state and /usr/sbin/alsactl).
#               Entries are whitespace-separated.
################################################################################

rtc

# End /etc/sysconfig/udev_retry
```

D.24. /sbin/ifup

```
#!/bin/sh
################################################################################
# Begin /sbin/ifup
#
# Description : Interface Up
#
# Authors     : Nathan Coulson - nathan@linuxfromscratch.org
```

```
#               Kevin P. Fleming - kpfleming@linuxfromscratch.org
# Update      : Bruce Dubbs - bdubbs@linuxfromscratch.org
#               DJ Lucas - dj@linuxfromscratch.org
#
# Version     : LFS 7.7
#
# Notes       : The IFCONFIG variable is passed to the SERVICE script
#               in the /lib/services directory, to indicate what file the
#               service should source to get interface specifications.
#
########################################################################

up()
{
  log_info_msg "Bringing up the ${1} interface..."

  if ip link show $1 > /dev/null 2>&1; then
     link_status=`ip link show $1`

     if [ -n "${link_status}" ]; then
        if ! echo "${link_status}" | grep -q UP; then
           ip link set $1 up
        fi
     fi

  else
     log_failure_msg "Interface ${IFACE} doesn't exist."
     exit 1
  fi

  evaluate_retval
}

RELEASE="7.7"

USAGE="Usage: $0 [ -hV ] [--help] [--version] interface"
VERSTR="LFS ifup, version ${RELEASE}"

while [ $# -gt 0 ]; do
   case "$1" in
      --help | -h)     help="y"; break ;;

      --version | -V)  echo "${VERSTR}"; exit 0 ;;

      -*)              echo "ifup: ${1}: invalid option" >&2
                       echo "${USAGE}" >& 2
                       exit 2 ;;

      *)               break ;;
   esac
done

if [ -n "$help" ]; then
   echo "${VERSTR}"
   echo "${USAGE}"
   echo
   cat << HERE_EOF
ifup is used to bring up a network interface.  The interface
parameter, e.g. eth0 or eth0:2, must match the trailing part of the
interface specifications file, e.g. /etc/sysconfig/ifconfig.eth0:2.

HERE_EOF
   exit 0
fi

file=/etc/sysconfig/ifconfig.${1}
```

```
# Skip backup files
[ "${file}" = "${file%""~""}" ] || exit 0

. /lib/lsb/init-functions

if [ ! -r "${file}" ]; then
   log_failure_msg "Unable to bring up ${1} interface! ${file} is missing or cannot be accessed."
   exit 1
fi

.  $file

if [ "$IFACE" = "" ]; then
   log_failure_msg "Unable to bring up ${1} interface! ${file} does not define an interface [IFACE]."
   exit 1
fi

# Do not process this service if started by boot, and ONBOOT
# is not set to yes
if [ "${IN_BOOT}" = "1" -a "${ONBOOT}" != "yes" ]; then
   exit 0
fi

# Bring up the interface
if [ "$VIRTINT" != "yes" ]; then
   up ${IFACE}
fi

for S in ${SERVICE}; do
  if [ ! -x "/lib/services/${S}" ]; then
    MSG="\nUnable to process ${file}.  Either "
    MSG="${MSG}the SERVICE '${S} was not present "
    MSG="${MSG}or cannot be executed."
    log_failure_msg "$MSG"
    exit 1
  fi
done

if [ "${SERVICE}" = "wpa" ]; then log_success_msg; fi

# Create/configure the interface
for S in ${SERVICE}; do
  IFCONFIG=${file} /lib/services/${S} ${IFACE} up
done

# Set link up virtual interfaces
if [ "${VIRTINT}" == "yes" ]; then
   up ${IFACE}
fi

# Bring up any additional interface components
for I in $INTERFACE_COMPONENTS; do up $I; done

# Set MTU if requested. Check if MTU has a "good" value.
if test -n "${MTU}"; then
   if [[ ${MTU} =~ ^[0-9]+$ ]] && [[ $MTU -ge 68 ]] ; then
      for I in $IFACE $INTERFACE_COMPONENTS; do
         ip link set dev $I mtu $MTU;
      done
   else
      log_info_msg2 "Invalid MTU $MTU"
   fi
fi

# Set the route default gateway if requested
if [ -n "${GATEWAY}" ]; then
   if ip route | grep -q default; then
```

```
      log_warning_msg "Gateway already setup; skipping."
   else
      log_info_msg "Adding default gateway ${GATEWAY} to the ${IFACE} interface..."
      ip route add default via ${GATEWAY} dev ${IFACE}
      evaluate_retval
   fi
fi

# End /sbin/ifup
```

D.25. /sbin/ifdown

```
#!/bin/bash
########################################################################
# Begin /sbin/ifdown
#
# Description : Interface Down
#
# Authors     : Nathan Coulson - nathan@linuxfromscratch.org
#               Kevin P. Fleming - kpfleming@linuxfromscratch.org
# Update      : Bruce Dubbs - bdubbs@linuxfromscratch.org
#
# Version     : LFS 7.0
#
# Notes       : the IFCONFIG variable is passed to the scripts found
#               in the /lib/services directory, to indicate what file the
#               service should source to get interface specifications.
#
########################################################################

RELEASE="7.0"

USAGE="Usage: $0 [ -hV ] [--help] [--version] interface"
VERSTR="LFS ifdown, version ${RELEASE}"

while [ $# -gt 0 ]; do
   case "$1" in
      --help | -h)     help="y"; break ;;

      --version | -V)  echo "${VERSTR}"; exit 0 ;;

      -*)              echo "ifup: ${1}: invalid option" >&2
                       echo "${USAGE}" >& 2
                       exit 2 ;;

      *)               break ;;
   esac
done

if [ -n "$help" ]; then
   echo "${VERSTR}"
   echo "${USAGE}"
   echo
   cat << HERE_EOF
ifdown is used to bring down a network interface.  The interface
parameter, e.g. eth0 or eth0:2, must match the trailing part of the
interface specifications file, e.g. /etc/sysconfig/ifconfig.eth0:2.

HERE_EOF
   exit 0
fi

file=/etc/sysconfig/ifconfig.${1}

# Skip backup files
[ "${file}" = "${file%""~""}" ] || exit 0
```

```
. /lib/lsb/init-functions

if [ ! -r "${file}" ]; then
   log_warning_msg "${file} is missing or cannot be accessed."
   exit 1
fi

. ${file}

if [ "$IFACE" = "" ]; then
   log_failure_msg "${file} does not define an interface [IFACE]."
   exit 1
fi

# We only need to first service to bring down the interface
S=`echo ${SERVICE} | cut -f1 -d" "`

if ip link show ${IFACE} > /dev/null 2>&1; then
   if [ -n "${S}" -a -x "/lib/services/${S}" ]; then
     IFCONFIG=${file} /lib/services/${S} ${IFACE} down
   else
     MSG="Unable to process ${file}.  Either "
     MSG="${MSG}the SERVICE variable was not set "
     MSG="${MSG}or the specified service cannot be executed."
     log_failure_msg "$MSG"
     exit 1
  fi
else
   log_warning_msg "Interface ${1} doesn't exist."
fi

# Leave the interface up if there are additional interfaces in the device
link_status=`ip link show ${IFACE} 2>/dev/null`

if [ -n "${link_status}" ]; then
   if [ "$(echo "${link_status}" | grep UP)" != "" ]; then
      if [ "$(ip addr show ${IFACE} | grep 'inet ')" == ""  ]; then
         log_info_msg "Bringing down the ${IFACE} interface..."
         ip link set ${IFACE} down
         evaluate_retval
      fi
   fi
fi

# End /sbin/ifdown
```

D.26. /lib/services/ipv4-static

```
#!/bin/sh
########################################################################
# Begin /lib/services/ipv4-static
#
# Description : IPV4 Static Boot Script
#
# Authors     : Nathan Coulson - nathan@linuxfromscratch.org
#               Kevin P. Fleming - kpfleming@linuxfromscratch.org
# Update      : Bruce Dubbs - bdubbs@linuxfromscratch.org
#
# Version     : LFS 7.0
#
########################################################################

. /lib/lsb/init-functions
. ${IFCONFIG}

if [ -z "${IP}" ]; then
   log_failure_msg "\nIP variable missing from ${IFCONFIG}, cannot continue."
```

```
   exit 1
fi

if [ -z "${PREFIX}" -a -z "${PEER}" ]; then
   log_warning_msg "\nPREFIX variable missing from ${IFCONFIG}, assuming 24."
   PREFIX=24
   args="${args} ${IP}/${PREFIX}"

elif [ -n "${PREFIX}" -a -n "${PEER}" ]; then
   log_failure_msg "\nPREFIX and PEER both specified in ${IFCONFIG}, cannot continue."
   exit 1

elif [ -n "${PREFIX}" ]; then
   args="${args} ${IP}/${PREFIX}"

elif [ -n "${PEER}" ]; then
   args="${args} ${IP} peer ${PEER}"
fi

if [ -n "${LABEL}" ]; then
   args="${args} label ${LABEL}"
fi

if [ -n "${BROADCAST}" ]; then
   args="${args} broadcast ${BROADCAST}"
fi

case "${2}" in
   up)
      if [ "$(ip addr show ${1} 2>/dev/null | grep ${IP}/)" = "" ]; then
         log_info_msg "Adding IPv4 address ${IP} to the ${1} interface..."
         ip addr add ${args} dev ${1}
         evaluate_retval
      else
         log_warning_msg "Cannot add IPv4 address ${IP} to ${1}.  Already present."
      fi
   ;;

   down)
      if [ "$(ip addr show ${1} 2>/dev/null | grep ${IP}/)" != "" ]; then
         log_info_msg "Removing IPv4 address ${IP} from the ${1} interface..."
         ip addr del ${args} dev ${1}
         evaluate_retval
      fi

      if [ -n "${GATEWAY}" ]; then
         # Only remove the gateway if there are no remaining ipv4 addresses
         if [ "$(ip addr show ${1} 2>/dev/null | grep 'inet ')" != "" ]; then
            log_info_msg "Removing default gateway..."
            ip route del default
            evaluate_retval
         fi
      fi
   ;;

   *)
      echo "Usage: ${0} [interface] {up|down}"
      exit 1
   ;;
esac

# End /lib/services/ipv4-static
```

D.27. /lib/services/ipv4-static-route

```
#!/bin/sh
########################################################################
```

```
# Begin /lib/services/ipv4-static-route
#
# Description : IPV4 Static Route Script
#
# Authors     : Kevin P. Fleming - kpfleming@linuxfromscratch.org
#               DJ Lucas - dj@linuxfromscratch.org
# Update      : Bruce Dubbs - bdubbs@linuxfromscratch.org
#
# Version     : LFS 7.0
#
########################################################################

. /lib/lsb/init-functions
. ${IFCONFIG}

case "${TYPE}" in
   ("" | "network")
      need_ip=1
      need_gateway=1
   ;;

   ("default")
      need_gateway=1
      args="${args} default"
      desc="default"
   ;;

   ("host")
      need_ip=1
   ;;

   ("unreachable")
      need_ip=1
      args="${args} unreachable"
      desc="unreachable "
   ;;

   (*)
      log_failure_msg "Unknown route type (${TYPE}) in ${IFCONFIG}, cannot continue."
      exit 1
   ;;
esac

if [ -n "${GATEWAY}" ]; then
   MSG="The GATEWAY variable cannot be set in ${IFCONFIG} for static routes.\n"
   log_failure_msg "$MSG Use STATIC_GATEWAY only, cannot continue"
   exit 1
fi

if [ -n "${need_ip}" ]; then
   if [ -z "${IP}" ]; then
      log_failure_msg "IP variable missing from ${IFCONFIG}, cannot continue."
      exit 1
   fi

   if [ -z "${PREFIX}" ]; then
      log_failure_msg "PREFIX variable missing from ${IFCONFIG}, cannot continue."
      exit 1
   fi

   args="${args} ${IP}/${PREFIX}"
   desc="${desc}${IP}/${PREFIX}"
fi

if [ -n "${need_gateway}" ]; then
   if [ -z "${STATIC_GATEWAY}" ]; then
      log_failure_msg "STATIC_GATEWAY variable missing from ${IFCONFIG}, cannot continue."
```

```
      exit 1
   fi
   args="${args} via ${STATIC_GATEWAY}"
fi

if [ -n "${SOURCE}" ]; then
        args="${args} src ${SOURCE}"
fi

case "${2}" in
   up)
      log_info_msg "Adding '${desc}' route to the ${1} interface..."
      ip route add ${args} dev ${1}
      evaluate_retval
   ;;

   down)
      log_info_msg "Removing '${desc}' route from the ${1} interface..."
      ip route del ${args} dev ${1}
      evaluate_retval
   ;;

   *)
      echo "Usage: ${0} [interface] {up|down}"
      exit 1
   ;;
esac

# End /lib/services/ipv4-static-route
```

Annexe E. Règles de configuration d'Udev

Les règles de cette annexe sont listées pour votre commodité. Leur installation se fait en principe via les instructions du Section 8.70, « Eudev-3.2.11 ».

E.1. 55-lfs.rules

```
# /etc/udev/rules.d/55-lfs.rules: Rule definitions for LFS.

# Core kernel devices

# This causes the system clock to be set as soon as /dev/rtc becomes available.
SUBSYSTEM=="rtc", ACTION=="add", MODE="0644", RUN+="/etc/rc.d/init.d/setclock start"
KERNEL=="rtc", ACTION=="add", MODE="0644", RUN+="/etc/rc.d/init.d/setclock start"

# Comms devices

KERNEL=="ippp[0-9]*",       GROUP="dialout"
KERNEL=="isdn[0-9]*",       GROUP="dialout"
KERNEL=="isdnctrl[0-9]*",   GROUP="dialout"
KERNEL=="dcbri[0-9]*",      GROUP="dialout"
```

Annexe F. 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.

F.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/*.

F.2. The MIT License

Copyright © 1999-2022 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: 121
Attr: 120
Autoconf: 156
Automake: 158
Bash: 142
  outils: 54
Bash: 142
  outils: 54
Bc: 107
Binutils: 113
  outils, passe 1: 41
  outils, passe 2: 67
Binutils: 113
  outils, passe 1: 41
  outils, passe 2: 67
Binutils: 113
  outils, passe 1: 41
  outils, passe 2: 67
Bison: 140
  outils: 76
Bison: 140
  outils: 76
Bootscripts: 220
  utilisation: 230
Bootscripts: 220
  utilisation: 230
Bzip2: 99
Check: 176
Coreutils: 171
  outils: 55
Coreutils: 171
  outils: 55
DejaGNU: 112
Diffutils: 177
  outils: 56
Diffutils: 177
  outils: 56
E2fsprogs: 211
Eudev: 199
  configuration: 199
Eudev: 199
  configuration: 199
Expat: 147
Expect: 111
File: 104
  outils: 57
File: 104
  outils: 57
Findutils: 179
  outils: 58
Findutils: 179
  outils: 58
Flex: 108
Gawk: 178
  outils: 59
Gawk: 178
  outils: 59
GCC: 127
  outils, libstdc++ passe 1: 49
  outils, passe 1: 43
  outils, passe 2: 68
GCC: 127
  outils, libstdc++ passe 1: 49
  outils, passe 1: 43
  outils, passe 2: 68
GCC: 127
  outils, libstdc++ passe 1: 49
  outils, passe 1: 43
  outils, passe 2: 68
GCC: 127
  outils, libstdc++ passe 1: 49
  outils, passe 1: 43
  outils, passe 2: 68
GDBM: 145
Gettext: 138
  outils: 75
Gettext: 138
  outils: 75
Glibc: 91
  outils: 46
Glibc: 91
  outils: 46
GMP: 116
Gperf: 146
Grep: 141
  outils: 60
Grep: 141
  outils: 60
Groff: 180
GRUB: 183
Gzip: 185
  outils: 61
Gzip: 185
  outils: 61
Iana-Etc: 90
Inetutils: 148

Intltool: 155
IPRoute2: 186
Kbd: 188
Kmod: 161
Less: 150
Libcap: 122
Libelf: 163
libffi: 164
Libpipeline: 190
Libtool: 144
Linux: 245
  outils, en-têtes API: 45
Linux: 245
  outils, en-têtes API: 45
M4: 106
  outils: 51
M4: 106
  outils: 51
Make: 191
  outils: 62
Make: 191
  outils: 62
Man-DB: 201
Man-pages: 89
Meson: 170
MPC: 119
MPFR: 118
Ncurses: 133
  outils: 52
Ncurses: 133
  outils: 52
Ninja: 168
OpenSSL: 159
Patch: 192
  outils: 63
Patch: 192
  outils: 63
Perl: 151
  outils: 77
Perl: 151
  outils: 77
Pkgconfig: 132
Procps-ng: 204
Psmisc: 137
Python: 165
  temporary: 78
Python: 165
  temporary: 78
rc.site: 236
Readline: 105
Sed: 136
  outils: 64
Sed: 136
  outils: 64
Shadow: 123
  configuration: 124
Shadow: 123
  configuration: 124
Sysklogd: 214
  configuration: 214
Sysklogd: 214
  configuration: 214
Sysvinit: 215
  configuration: 231
Sysvinit: 215
  configuration: 231
Tar: 193
  outils: 65
Tar: 193
  outils: 65
Tcl: 109
Texinfo: 194
  temporary: 79
Texinfo: 194
  temporary: 79
Udev
  utilisation: 222
Util-linux: 206
  outils: 80
Util-linux: 206
  outils: 80
Vim: 196
wheel: 167
XML::Parser: 154
Xz: 101
  outils: 66
Xz: 101
  outils: 66
Zlib: 98
zstd: 103

Programmes

[: 171, 172
2to3: 165
accessdb: 201, 202
aclocal: 158, 158
aclocal-1.16: 158, 158
addftinfo: 180, 180
addpart: 206, 207
addr2line: 113, 114

afmtodit: 180, 180
agetty: 206, 207
apropos: 201, 202
ar: 113, 114
as: 113, 114
attr: 120, 120
autoconf: 156, 156
autoheader: 156, 156
autom4te: 156, 156
automake: 158, 158
automake-1.16: 158, 158
autopoint: 138, 138
autoreconf: 156, 156
autoscan: 156, 156
autoupdate: 156, 156
awk: 178, 178
b2sum: 171, 172
badblocks: 211, 212
base64: 171, 172, 171, 172
base64: 171, 172, 171, 172
basename: 171, 172
basenc: 171, 172
bash: 142, 142
bashbug: 142, 143
bc: 107, 107
bison: 140, 140
blkdiscard: 206, 207
blkid: 206, 207
blkzone: 206, 207
blockdev: 206, 207
bootlogd: 215, 215
bridge: 186, 186
bunzip2: 99, 100
bzcat: 99, 100
bzcmp: 99, 100
bzdiff: 99, 100
bzegrep: 99, 100
bzfgrep: 99, 100
bzgrep: 99, 100
bzip2: 99, 100
bzip2recover: 99, 100
bzless: 99, 100
bzmore: 99, 100
c++: 127, 130
c++filt: 113, 114
cal: 206, 207
capsh: 122, 122
captoinfo: 133, 134
cat: 171, 172
catman: 201, 203
cc: 127, 130
cfdisk: 206, 207
chacl: 121, 121
chage: 123, 125
chattr: 211, 212
chcon: 171, 172
chcpu: 206, 207
checkmk: 176, 176
chem: 180, 180
chfn: 123, 125
chgpasswd: 123, 125
chgrp: 171, 172
chmem: 206, 207
chmod: 171, 172
choom: 206, 207
chown: 171, 172
chpasswd: 123, 125
chroot: 171, 172
chrt: 206, 207
chsh: 123, 125
chvt: 188, 189
cksum: 171, 172
clear: 133, 134
cmp: 177, 177
col: 206, 207
colcrt: 206, 207
colrm: 206, 207
column: 206, 207
comm: 171, 172
compile_et: 211, 212
corelist: 151, 152
cp: 171, 173
cpan: 151, 152
cpp: 127, 130
csplit: 171, 173
ctrlaltdel: 206, 207
ctstat: 186, 186
cut: 171, 173
c_rehash: 159, 160
date: 171, 173
dc: 107, 107
dd: 171, 173
deallocvt: 188, 189
debugfs: 211, 212
dejagnu: 112, 112
delpart: 206, 207
depmod: 161, 161
df: 171, 173
diff: 177, 177
diff3: 177, 177

dir: 171, 173
dircolors: 171, 173
dirname: 171, 173
dmesg: 206, 207
dnsdomainname: 148, 149
du: 171, 173
dumpe2fs: 211, 212
dumpkeys: 188, 189
e2freefrag: 211, 212
e2fsck: 211, 212
e2image: 211, 212
e2label: 211, 212
e2mmpstatus: 211, 212
e2scrub: 211, 212
e2scrub_all: 211, 212
e2undo: 211, 212
e4crypt: 211, 212
e4defrag: 211, 212
echo: 171, 173
egrep: 141, 141
eject: 206, 207
elfedit: 113, 114
enc2xs: 151, 152
encguess: 151, 152
env: 171, 173
envsubst: 138, 138
eqn: 180, 180
eqn2graph: 180, 180
ex: 196, 198
expand: 171, 173
expect: 111, 111
expiry: 123, 125
expr: 171, 173
factor: 171, 173
faillog: 123, 125
fallocate: 206, 207
false: 171, 173
fdisk: 206, 207
fgconsole: 188, 189
fgrep: 141, 141
file: 104, 104
filefrag: 211, 212
fincore: 206, 208
find: 179, 179
findfs: 206, 208
findmnt: 206, 208
flex: 108, 108
flex++: 108, 108
flock: 206, 208
fmt: 171, 173
fold: 171, 173
free: 204, 204
fsck: 206, 208
fsck.cramfs: 206, 208
fsck.ext2: 211, 212
fsck.ext3: 211, 212
fsck.ext4: 211, 212
fsck.minix: 206, 208
fsfreeze: 206, 208
fstab-decode: 215, 215
fstrim: 206, 208
ftp: 148, 149
fuser: 137, 137
g++: 127, 130
gawk: 178, 178
gawk-5.1.1: 178, 178
gcc: 127, 130
gc-ar: 127, 130
gc-nm: 127, 130
gc-ranlib: 127, 130
gcov: 127, 130
gcov-dump: 127, 130
gcov-tool: 127, 130
gdbmtool: 145, 145
gdbm_dump: 145, 145
gdbm_load: 145, 145
gdiffmk: 180, 180
gencat: 91, 96
genl: 186, 186
getcap: 122, 122
getconf: 91, 96
getent: 91, 96
getfacl: 121, 121
getfattr: 120, 120
getkeycodes: 188, 189
getopt: 206, 208
getpcaps: 122, 122
getsubids: 123, 125
gettext: 138, 138
gettext.sh: 138, 138
gettextize: 138, 138
glilypond: 180, 180
gpasswd: 123, 125
gperf: 146, 146
gperl: 180, 180
gpinyin: 180, 180
gprof: 113, 114
grap2graph: 180, 180
grep: 141, 141
grn: 180, 181

grodvi: 180, 181
groff: 180, 181
groffer: 180, 181
grog: 180, 181
grolbp: 180, 181
grolj4: 180, 181
gropdf: 180, 181
grops: 180, 181
grotty: 180, 181
groupadd: 123, 125
groupdel: 123, 125
groupmems: 123, 125
groupmod: 123, 125
groups: 171, 173
grpck: 123, 125
grpconv: 123, 125
grpunconv: 123, 125
grub-bios-setup: 183, 183
grub-editenv: 183, 184
grub-file: 183, 184
grub-fstest: 183, 184
grub-glue-efi: 183, 184
grub-install: 183, 184
grub-kbdcomp: 183, 184
grub-macbless: 183, 184
grub-menulst2cfg: 183, 184
grub-mkconfig: 183, 184
grub-mkimage: 183, 184
grub-mklayout: 183, 184
grub-mknetdir: 183, 184
grub-mkpasswd-pbkdf2: 183, 184
grub-mkrelpath: 183, 184
grub-mkrescue: 183, 184
grub-mkstandalone: 183, 184
grub-ofpathname: 183, 184
grub-probe: 183, 184
grub-reboot: 183, 184
grub-render-label: 183, 184
grub-script-check: 183, 184
grub-set-default: 183, 184
grub-setup: 183, 184
grub-syslinux2cfg: 183, 184
gunzip: 185, 185
gzexe: 185, 185
gzip: 185, 185
h2ph: 151, 152
h2xs: 151, 152
halt: 215, 215
head: 171, 173
hexdump: 206, 208
hostid: 171, 173
hostname: 148, 149
hpftodit: 180, 181
hwclock: 206, 208
i386: 206, 208
iconv: 91, 96
iconvconfig: 91, 96
id: 171, 173
idle3: 165
ifcfg: 186, 186
ifconfig: 148, 149
ifnames: 156, 156
ifstat: 186, 186
indxbib: 180, 181
info: 194, 195
infocmp: 133, 134
infotocap: 133, 134
init: 215, 215
insmod: 161, 161
install: 171, 173
install-info: 194, 195
instmodsh: 151, 152
intltool-extract: 155, 155
intltool-merge: 155, 155
intltool-prepare: 155, 155
intltool-update: 155, 155
intltoolize: 155, 155
ionice: 206, 208
ip: 186, 186
ipcmk: 206, 208
ipcrm: 206, 208
ipcs: 206, 208
irqtop: 206, 208
isosize: 206, 208
join: 171, 173
json_pp: 151, 152
kbdinfo: 188, 189
kbdrate: 188, 189
kbd_mode: 188, 189
kill: 206, 208
killall: 137, 137
killall5: 215, 215
klogd: 214, 214
kmod: 161, 161
last: 206, 208
lastb: 206, 208
lastlog: 123, 125
ld: 113, 114
ld.bfd: 113, 115
ld.gold: 113, 115

ldattach: 206, 208
ldconfig: 91, 96
ldd: 91, 96
lddlibc4: 91, 96
less: 150, 150
lessecho: 150, 150
lesskey: 150, 150
lex: 108, 108
lexgrog: 201, 203
lfskernel-5.18.3: 245, 248
libasan: 127, 130
libatomic: 127, 130
libcc1: 127, 130
libnetcfg: 151, 152
libtool: 144, 144
libtoolize: 144, 144
link: 171, 173
linux32: 206, 208
linux64: 206, 208
lkbib: 180, 181
ln: 171, 173
lnstat: 186, 187
loadkeys: 188, 189
loadunimap: 188, 189
locale: 91, 96
localedef: 91, 96
locate: 179, 179
logger: 206, 208
login: 123, 125
logname: 171, 173
logoutd: 123, 125
logsave: 211, 212
look: 206, 208
lookbib: 180, 181
losetup: 206, 208
ls: 171, 173
lsattr: 211, 212
lsblk: 206, 208
lscpu: 206, 208
lsipc: 206, 208
lsirq: 206, 208
lslocks: 206, 208
lslogins: 206, 208
lsmem: 206, 209
lsmod: 161, 161
lsns: 206, 209
lto-dump: 127, 130
lzcat: 101, 101
lzcmp: 101, 101
lzdiff: 101, 101
lzegrep: 101, 101
lzfgrep: 101, 101
lzgrep: 101, 101
lzless: 101, 101
lzma: 101, 101
lzmadec: 101, 101
lzmainfo: 101, 102
lzmore: 101, 102
m4: 106, 106
make: 191, 191
makedb: 91, 96
makeinfo: 194, 195
man: 201, 203
man-recode: 201, 203
mandb: 201, 203
manpath: 201, 203
mapscrn: 188, 189
mcookie: 206, 209
md5sum: 171, 173
mesg: 206, 209
meson: 170, 170
mkdir: 171, 173
mke2fs: 211, 212
mkfifo: 171, 173
mkfs: 206, 209
mkfs.bfs: 206, 209
mkfs.cramfs: 206, 209
mkfs.ext2: 211, 212
mkfs.ext3: 211, 212
mkfs.ext4: 211, 213
mkfs.minix: 206, 209
mklost+found: 211, 213
mknod: 171, 173
mkswap: 206, 209
mktemp: 171, 174
mk_cmds: 211, 212
mmroff: 180, 181
modinfo: 161, 162
modprobe: 161, 162
more: 206, 209
mount: 206, 209
mountpoint: 206, 209
msgattrib: 138, 138
msgcat: 138, 138
msgcmp: 138, 138
msgcomm: 138, 139
msgconv: 138, 139
msgen: 138, 139
msgexec: 138, 139
msgfilter: 138, 139

msgfmt: 138, 139
msggrep: 138, 139
msginit: 138, 139
msgmerge: 138, 139
msgunfmt: 138, 139
msguniq: 138, 139
mtrace: 91, 96
mv: 171, 174
namei: 206, 209
ncursesw6-config: 133, 134
neqn: 180, 181
newgidmap: 123, 125
newgrp: 123, 125
newuidmap: 123, 125
newusers: 123, 125
ngettext: 138, 139
nice: 171, 174
ninja: 168, 169
nl: 171, 174
nm: 113, 115
nohup: 171, 174
nologin: 123, 125
nproc: 171, 174
nroff: 180, 181
nscd: 91, 96
nsenter: 206, 209
nstat: 186, 187
numfmt: 171, 174
objcopy: 113, 115
objdump: 113, 115
od: 171, 174
openssl: 159, 160
openvt: 188, 189
partx: 206, 209
passwd: 123, 126
paste: 171, 174
patch: 192, 192
pathchk: 171, 174
pcprofiledump: 91, 96
pdfmom: 180, 181
pdfroff: 180, 181
pdftexi2dvi: 194, 195
peekfd: 137, 137
perl: 151, 152
perl5.36.0: 151, 152
perlbug: 151, 152
perldoc: 151, 152
perlivp: 151, 152
perlthanks: 151, 152
pfbtops: 180, 181
pgrep: 204, 204
pic: 180, 181
pic2graph: 180, 181
piconv: 151, 152
pidof: 204, 204
ping: 148, 149
ping6: 148, 149
pinky: 171, 174
pip3: 165
pivot_root: 206, 209
pkg-config: 132, 132
pkill: 204, 204
pl2pm: 151, 152
pldd: 91, 96
pmap: 204, 204
pod2html: 151, 153
pod2man: 151, 153
pod2texi: 194, 195
pod2text: 151, 153
pod2usage: 151, 153
podchecker: 151, 153
podselect: 151, 153
post-grohtml: 180, 181
poweroff: 215, 215
pr: 171, 174
pre-grohtml: 180, 181
preconv: 180, 181
printenv: 171, 174
printf: 171, 174
prlimit: 206, 209
prove: 151, 153
prtstat: 137, 137
ps: 204, 204
psfaddtable: 188, 189
psfgettable: 188, 189
psfstriptable: 188, 189
psfxtable: 188, 189
pslog: 137, 137
pstree: 137, 137
pstree.x11: 137, 137
ptar: 151, 153
ptardiff: 151, 153
ptargrep: 151, 153
ptx: 171, 174
pwait: 204, 204
pwck: 123, 126
pwconv: 123, 126
pwd: 171, 174
pwdx: 204, 204
pwunconv: 123, 126

pydoc3: 165
python3: 165
ranlib: 113, 115
readelf: 113, 115
readlink: 171, 174
readprofile: 206, 209
realpath: 171, 174
reboot: 215, 215
recode-sr-latin: 138, 139
refer: 180, 181
rename: 206, 209
renice: 206, 209
reset: 133, 134
resize2fs: 211, 213
resizepart: 206, 209
rev: 206, 209
rkfill: 206, 209
rm: 171, 174
rmdir: 171, 174
rmmod: 161, 162
roff2dvi: 180, 182
roff2html: 180, 182
roff2pdf: 180, 182
roff2ps: 180, 182
roff2text: 180, 182
roff2x: 180, 182
routef: 186, 187
routel: 186, 187
rtacct: 186, 187
rtcwake: 206, 209
rtmon: 186, 187
rtpr: 186, 187
rtstat: 186, 187
runcon: 171, 174
runlevel: 215, 215
runtest: 112, 112
rview: 196, 198
rvim: 196, 198
script: 206, 209
scriptlive: 206, 209
scriptreplay: 206, 209
sdiff: 177, 177
sed: 136, 136
seq: 171, 174
setarch: 206, 209
setcap: 122, 122
setfacl: 121, 121
setfattr: 120, 120
setfont: 188, 189
setkeycodes: 188, 189
setleds: 188, 189
setmetamode: 188, 189
setsid: 206, 209
setterm: 206, 209
setvtrgb: 188, 189
sfdisk: 206, 209
sg: 123, 126
sh: 142, 143
sha1sum: 171, 174
sha224sum: 171, 174
sha256sum: 171, 174
sha384sum: 171, 174
sha512sum: 171, 174
shasum: 151, 153
showconsolefont: 188, 189
showkey: 188, 189
shred: 171, 174
shuf: 171, 174
shutdown: 215, 215
size: 113, 115
slabtop: 204, 204
sleep: 171, 174
sln: 91, 96
soelim: 180, 182
sort: 171, 174
sotruss: 91, 96
splain: 151, 153
split: 171, 174
sprof: 91, 96
ss: 186, 187
stat: 171, 174
stdbuf: 171, 174
strings: 113, 115
strip: 113, 115
stty: 171, 174
su: 123, 126
sulogin: 206, 209
sum: 171, 174
swaplabel: 206, 209
swapoff: 206, 209
swapon: 206, 210
switch_root: 206, 210
sync: 171, 175
sysctl: 204, 205
syslogd: 214, 214
tabs: 133, 134
tac: 171, 175
tail: 171, 175
talk: 148, 149
tar: 193, 193

taskset: 206, 210
tbl: 180, 182
tc: 186, 187
tclsh: 109, 110
tclsh8.6: 109, 110
tee: 171, 175
telinit: 215, 215
telnet: 148, 149
test: 171, 175
texi2dvi: 194, 195
texi2pdf: 194, 195
texi2any: 194, 195
texindex: 194, 195
tfmtodit: 180, 182
tftp: 148, 149
tic: 133, 134
timeout: 171, 175
tload: 204, 205
toe: 133, 134
top: 204, 205
touch: 171, 175
tput: 133, 134
tr: 171, 175
traceroute: 148, 149
troff: 180, 182
true: 171, 175
truncate: 171, 175
tset: 133, 135
tsort: 171, 175
tty: 171, 175
tune2fs: 211, 213
tzselect: 91, 96
uclampset: 206, 210
udevadm: 199, 199
udevd: 199, 200
ul: 206, 210
umount: 206, 210
uname: 171, 175
uname26: 206, 210
uncompress: 185, 185
unexpand: 171, 175
unicode_start: 188, 189
unicode_stop: 188, 189
uniq: 171, 175
unlink: 171, 175
unlzma: 101, 102
unshare: 206, 210
unxz: 101, 102
updatedb: 179, 179
uptime: 204, 205
useradd: 123, 126
userdel: 123, 126
usermod: 123, 126
users: 171, 175
utmpdump: 206, 210
uuidd: 206, 210
uuidgen: 206, 210
uuidparse: 206, 210
vdir: 171, 175
vi: 196, 198
view: 196, 198
vigr: 123, 126
vim: 196, 198
vimdiff: 196, 198
vimtutor: 196, 198
vipw: 123, 126
vmstat: 204, 205
w: 204, 205
wall: 206, 210
watch: 204, 205
wc: 171, 175
wdctl: 206, 210
whatis: 201, 203
wheel: 167
whereis: 206, 210
who: 171, 175
whoami: 171, 175
wipefs: 206, 210
x86_64: 206, 210
xargs: 179, 179
xgettext: 138, 139
xmlwf: 147, 147
xsubpp: 151, 153
xtrace: 91, 96
xxd: 196, 198
xz: 101, 102
xzcat: 101, 102
xzcmp: 101, 102
xzdec: 101, 102
xzdiff: 101, 102
xzegrep: 101, 102
xzfgrep: 101, 102
xzgrep: 101, 102
xzless: 101, 102
xzmore: 101, 102
yacc: 140, 140
yes: 171, 175
zcat: 185, 185
zcmp: 185, 185
zdiff: 185, 185

zdump: 91, 96
zegrep: 185, 185
zfgrep: 185, 185
zforce: 185, 185
zgrep: 185, 185
zic: 91, 96
zipdetails: 151, 153
zless: 185, 185
zmore: 185, 185
znew: 185, 185
zramctl: 206, 210
zstd: 103, 103
zstdgrep: 103, 103
zstdless: 103, 103

Bibliothèques

Expat: 154, 154
ld-2.35.so: 91, 96
libacl: 121, 121
libanl: 91, 96
libasprintf: 138, 139
libattr: 120, 120
libbfd: 113, 115
libblkid: 206, 210
libBrokenLocale: 91, 96
libbz2: 99, 100
libc: 91, 96
libcap: 122, 122
libcheck: 176, 176
libcom_err: 211, 213
libcrypt: 91, 96
libcrypto.so: 159, 160
libctf: 113, 115
libctf-nobfd: 113, 115
libcursesw: 133, 135
libc_malloc_debug: 91, 96
libdl: 91, 96
libe2p: 211, 213
libelf: 163, 163
libexpat: 147, 147
libexpect-5.45.4: 111, 111
libext2fs: 211, 213
libfdisk: 206, 210
libffi: 164
libfl: 108, 108
libformw: 133, 135
libg: 91, 96
libgcc: 127, 130
libgcov: 127, 130
libgdbm: 145, 145
libgdbm_compat: 145, 145
libgettextlib: 138, 139
libgettextpo: 138, 139
libgettextsrc: 138, 139
libgmp: 116, 117
libgmpxx: 116, 117
libgomp: 127, 130
libhistory: 105, 105
libitm: 127, 130
libkmod: 161
liblsan: 127, 130
libltdl: 144, 144
liblto_plugin: 127, 131
liblzma: 101, 102
libm: 91, 97
libmagic: 104, 104
libman: 201, 203
libmandb: 201, 203
libmcheck: 91, 97
libmemusage: 91, 97
libmenuw: 133, 135
libmount: 206, 210
libmpc: 119, 119
libmpfr: 118, 118
libmvec: 91, 97
libncursesw: 133, 135
libnsl: 91, 97
libnss_*: 91, 97
libopcodes: 113, 115
libpanelw: 133, 135
libpcprofile: 91, 97
libpipeline: 190
libproc-2: 204, 205
libpsx: 122, 122
libpthread: 91, 97
libquadmath: 127, 131
libreadline: 105, 105
libresolv: 91, 97
librt: 91, 97
libsmartcols: 206, 210
libss: 211, 213
libssl.so: 159, 160
libssp: 127, 131
libstdbuf: 171, 175
libstdc++: 127, 131
libstdc++fs: 127, 131
libsubid: 123, 126
libsupc++: 127, 131
libtcl8.6.so: 109, 110
libtclstub8.6.a: 109, 110

libtextstyle: 138, 139
libthread_db: 91, 97
libtsan: 127, 131
libubsan: 127, 131
libudev: 199, 200
libutil: 91, 97
libuuid: 206, 210
liby: 140, 140
libz: 98, 98
libzstd: 103, 103
preloadable_libintl: 138, 139

Scripts

checkfs: 220, 220
cleanfs: 220, 220
console: 220, 220
  configuration: 233
console: 220, 220
  configuration: 233
File creation at boot
  configuration: 236
functions: 220, 220
halt: 220, 220
hostname
  configuration: 229
ifdown: 220, 220
ifup: 220, 220
ipv4-static: 220, 221
localnet: 220, 220
  /etc/hosts: 229
localnet: 220, 220
  /etc/hosts: 229
modules: 220, 220
mountfs: 220, 220
mountkernfs: 220, 220
network: 220, 220
  /etc/hosts: 229
  configuration: 228
network: 220, 220
  /etc/hosts: 229
  configuration: 228
network: 220, 220
  /etc/hosts: 229
  configuration: 228
rc: 220, 220
reboot: 220, 220
sendsignals: 220, 220
setclock: 220, 221
  configuration: 233
setclock: 220, 221
  configuration: 233
swap: 220, 221
sysctl: 220, 221
sysklogd: 220, 221
  configuration: 236
sysklogd: 220, 221
  configuration: 236
template: 220, 221
udev: 220, 221
udev_retry: 220, 221
dwp: 113, 114

Autres

/boot/config-5.18.3: 245, 248
/boot/System.map-5.18.3: 245, 248
/dev/*: 70
/etc/fstab: 243
/etc/group: 72
/etc/hosts: 229
/etc/inittab: 231
/etc/inputrc: 240
/etc/ld.so.conf: 95
/etc/lfs-release: 251
/etc/localtime: 94
/etc/lsb-release: 251
/etc/modprobe.d/usb.conf: 248
/etc/nsswitch.conf: 94
/etc/os-release: 251
/etc/passwd: 72
/etc/profile: 238
/etc/protocols: 90
/etc/resolv.conf: 229
/etc/services: 90
/etc/syslog.conf: 214
/etc/udev: 199, 200
/etc/udev/hwdb.bin: 199
/etc/vimrc: 197
/run/utmp: 72
/usr/include/asm-generic/*.h: 45, 45
/usr/include/asm/*.h: 45, 45
/usr/include/drm/*.h: 45, 45
/usr/include/linux/*.h: 45, 45
/usr/include/misc/*.h: 45, 45
/usr/include/mtd/*.h: 45, 45
/usr/include/rdma/*.h: 45, 45
/usr/include/scsi/*.h: 45, 45
/usr/include/sound/*.h: 45, 45
/usr/include/video/*.h: 45, 45
/usr/include/xen/*.h: 45, 45
/var/log/btmp: 72

/var/log/lastlog: 72
/var/log/wtmp: 72
/etc/shells: 241
man pages: 89, 89