Linux From Scratch

Version 7.3

Créé par Gerard Beekmans
Édité par Matthew Burgess et Bruce Dubbs

Linux From Scratch: Version 7.3

par Créé par Gerard Beekmans et Édité par Matthew Burgess et Bruce Dubbs

Copyright © 1999-2012 Gerard Beekmans

Copyright © 1999-2012, 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. LFS et les standards x
v. Raison de la présence des paquets dans le livre xi
vi. Prérequis xvi
vii. Prérequis du système hôte xvii
viii. Typographie xix
ix. Structure xx
x. Errata xx
I. Introduction 1
1. Introduction 2
1.1. Comment construire un système LFS 2
1.2. Quoi de neuf depuis la dernière version 3
1.3. Historique des modifications 4
1.4. Ressources 9
1.5. Aide 10
II. Préparation à la construction 12
2. Préparer une nouvelle partition 13
2.1. Introduction 13
2.2. Créer une nouvelle partition 13
2.3. Créer un système de fichiers sur la partition 14
2.4. Monter la nouvelle partition 15
3. Paquets et correctifs 17
3.1. Introduction 17
3.2. Tous les paquets 17
3.3. Correctifs requis 24
4. Dernières préparations 25
4.1. À propos de $LFS 25
4.2. Créer le répertoire $LFS/tools 25
4.3. Ajouter l'utilisateur LFS 26
4.4. Configurer l'environnement 27
4.5. À propos des SBU 28
4.6. À propos des suites de tests 29
5. Construire un système temporaire 30
5.1. Introduction 30
5.2. Notes techniques sur la chaîne d'outils 30
5.3. Instructions générales de compilation 32
5.4. Binutils-2.23.1 - Passe 1 34
5.5. GCC-4.7.2 - Passe 1 36
5.6. Linux-3.8.1 API Headers 39
5.7. Glibc-2.17 40
5.8. Binutils-2.23.1 - Passe 2 43
5.9. GCC-4.7.2 - Passe 2 45

5.10. Tcl-8.6.0 49
5.11. Expect-5.45 51
5.12. DejaGNU-1.5 53
5.13. Check-0.9.9 54
5.14. Ncurses-5.9 55
5.15. Bash-4.2 56
5.16. Bzip2-1.0.6 57
5.17. Coreutils-8.21 58
5.18. Diffutils-3.2 59
5.19. File-5.13 60
5.20. Findutils-4.4.2 61
5.21. Gawk-4.0.2 62
5.22. Gettext-0.18.2 63
5.23. Grep-2.14 64
5.24. Gzip-1.5 65
5.25. M4-1.4.16 66
5.26. Make-3.82 67
5.27. Patch-2.7.1 68
5.28. Perl-5.16.2 69
5.29. Sed-4.2.2 70
5.30. Tar-1.26 71
5.31. Texinfo-5.0 72
5.32. Xz-5.0.4 73
5.33. Supprimer les symboles des fichiers objets 74
5.34. Changer de propriétaire 74
III. Construction du système LFS 75
6. Installer les logiciels du système de base 76
6.1. Introduction 76
6.2. Préparer les systèmes de fichiers virtuels du noyau 76
6.3. Gestion de paquets 77
6.4. Entrer dans l'environnement chroot 81
6.5. Créer les répertoires 82
6.6. Créer les fichiers et les liens symboliques essentiels 83
6.7. Linux-3.8.1 API Headers 86
6.8. Man-pages-3.46 88
6.9. Glibc-2.17 89
6.10. Ajustement de la chaîne d'outils 97
6.11. Zlib-1.2.7 99
6.12. File-5.13 100
6.13. Binutils-2.23.1 101
6.14. GMP-5.1.0 104
6.15. MPFR-3.1.1 106
6.16. MPC-1.0.1 107
6.17. GCC-4.7.2 108
6.18. Sed-4.2.2 114
6.19. Bzip2-1.0.6 115
6.20. Pkg-config-0.27.1 117

6.21. Ncurses-5.9 118
6.22. Util-linux-2.22.2 121
6.23. Psmisc-22.20 125
6.24. Procps-3.3.6 126
6.25. E2fsprogs-1.42.7 128
6.26. Shadow-4.1.5.1 131
6.27. Coreutils-8.21 135
6.28. Iana-Etc-2.30 140
6.29. M4-1.4.16 141
6.30. Bison-2.7 142
6.31. Grep-2.14 143
6.32. Readline-6.2 144
6.33. Bash-4.2 146
6.34. Libtool-2.4.2 148
6.35. GDBM-1.10 149
6.36. Inetutils-1.9.1 150
6.37. Perl-5.16.2 152
6.38. Autoconf-2.69 155
6.39. Automake-1.13.1 157
6.40. Diffutils-3.2 159
6.41. Gawk-4.0.2 160
6.42. Findutils-4.4.2 161
6.43. Flex-2.5.37 163
6.44. Gettext-0.18.2 165
6.45. Groff-1.22.1 167
6.46. Xz-5.0.4 170
6.47. GRUB-2.00 172
6.48. Less-451 174
6.49. Gzip-1.5 175
6.50. IPRoute2-3.8.0 177
6.51. Kbd-1.15.5 179
6.52. Kmod-12 182
6.53. Libpipeline-1.2.2 184
6.54. Make-3.82 185
6.55. Man-DB-2.6.3 186
6.56. Patch-2.7.1 189
6.57. Sysklogd-1.5 190
6.58. Sysvinit-2.88dsf 191
6.59. Tar-1.26 193
6.60. Texinfo-5.0 194
6.61. Udev-197 (Extrait de systemd-197) 196
6.62. Vim-7.3 198
6.63. À propos des symboles de débogage 201
6.64. Supprimer de nouveau les symboles des fichiers objets 201
6.65. Nettoyer 202
7. Initialiser les scripts de démarrage du système 203
7.1. Introduction 203

7.2. Configuration générale du réseau 203
7.3. Personnaliser le fichier /etc/hosts 206
7.4. Gestion des périphériques et modules d'un système LFS 207
7.5. Création de liens symboliques personnalisés vers les périphériques 211
7.6. LFS-Bootscripts-20130123 214
7.7. Comment fonctionnent ces scripts de démarrage ? 216
7.8. Configurer le nom d'hôte du système 218
7.9. Configurer le script setclock 219
7.10. Configurer la console Linux 219
7.11. Configurer le script sysklogd 222
7.12. Le fichier rc.site 223
7.13. Fichiers de démarrage du shell Bash 226
7.14. Créer le fichier /etc/inputrc 228
8. Rendre le système LFS amorçable 230
8.1. Introduction 230
8.2. Créer le fichier /etc/fstab 230
8.3. Linux-3.8.1 232
8.4. Utiliser GRUB pour paramétrer le processus de démarrage 235
9. Fin 237
9.1. Fin 237
9.2. Enregistrez-vous 237
9.3. Redémarrer le système 237
9.4. Et maintenant ? 239
IV. Annexes 241
A. Acronymes et Termes 242
B. Remerciements 245
C. Dépendances 248
D. Scripts de démarrage et de sysconfig version-20130123 264
D.1. /etc/rc.d/init.d/rc 264
D.2. /lib/lsb/init-functions 268
D.3. /etc/rc.d/init.d/functions 282
D.4. /etc/rc.d/init.d/mountvirtfs 296
D.5. /etc/rc.d/init.d/modules 297
D.6. /etc/rc.d/init.d/udev 299
D.7. /etc/rc.d/init.d/swap 300
D.8. /etc/rc.d/init.d/setclock 301
D.9. /etc/rc.d/init.d/checkfs 302
D.10. /etc/rc.d/init.d/mountfs 305
D.11. /etc/rc.d/init.d/udev_retry 307
D.12. /etc/rc.d/init.d/cleanfs 308
D.13. /etc/rc.d/init.d/console 310
D.14. /etc/rc.d/init.d/localnet 312
D.15. /etc/rc.d/init.d/sysctl 314
D.16. /etc/rc.d/init.d/sysklogd 315
D.17. /etc/rc.d/init.d/network 316
D.18. /etc/rc.d/init.d/sendsignals 318
D.19. /etc/rc.d/init.d/reboot 319

D.20. /etc/rc.d/init.d/halt 320
D.21. /etc/rc.d/init.d/template 320
D.22. /etc/sysconfig/modules 321
D.23. /etc/sysconfig/createfiles 322
D.24. /etc/sysconfig/udev-retry 322
D.25. /sbin/ifup 323
D.26. /sbin/ifdown 325
D.27. /lib/services/ipv4-static 327
D.28. /lib/services/ipv4-static-route 329
E. Règles de configuration d'Udev 331
E.1. 55-lfs.rules 331
F. Licences LFS 332
F.1. Creative Commons License 332
F.2. The MIT License 336
Index 337

Préface

Avant-propos

Mon cheminement pour apprendre et mieux comprendre Linux a commencé il y a plus d'une décennie, soit 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 plein de façons d'accomplir une seule tâche. Il en est de même pour les distributions Linux. Un grand nombre existent depuis des années. Certaines existent encore, d'autres se sont transformées en quelque chose d'autre, tandis que d'autres ont été reléguées dans nos souvenirs. Elles font toutes les choses différemment pour s'adapter au besoin de leurs destinataires. Vu qu'il existait énormément de manières différentes d'accomplir le même but final, je me rendis compte que je n'étais plus obligé de me limiter à une organisation en particulier. Avant de découvrir Linux, nous supportions simplement les problèmes dans d'autres systèmes d'exploitation puisque vous n'aviez pas le choix. Cela valait ce que ça valait, que vous aimiez ou pas. Avec Linux, le concept du choix a commencé à émerger. Si vous n'aimiez pas quelque chose, vous étiez libres, voire invités à le modifier.

J'ai essayé un certain nombre de distributions et n'ai pas pu me décider pour l'une d'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 parfait pour moi. Donc, je résolus de créer mon propre système Linux qui correspondrait complètement à mes préférences personnelles.

Pour réellement fabriquer 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. L'idée était au départ décourageante. Je restais sceptique à l'idée de pouvoir construire un tel système.

Après avoir rencontré quelques problèmes comme des dépendances circulaires et erreurs à la compilation, j'ai construit un système Linux entièrement personnalisé. Il était complètement opérationnel et parfaitement utilisable comme n'importe quel autre système Linux existant à ce moment. Mais c'était ma propre création. C'était très satisfaisant d'avoir concocté un système moi-même. Faire mieux aurait été de créer chaque morceau de logiciel moi-même. C'était le mieux qui pouvait suivre.

Lorsque j'ai partagé mes objectifs et mes expériences avec d'autres membres de la communauté Linux, il est devenu clair qu'il y avait un sérieux intérêt dans ces idées. Il devint rapidement clair que de tels systèmes LFS personnalisés rencontraient non seulement les exigences des utilisateurs mais servaient aussi comme opportunité idéale d'apprentissage pour les programmeurs et les administrateurs système, 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 principal de ce projet. Il fournit la base et les instructions qui vous sont nécessaires pour concevoir et créer votre propre système. Si ce livre fournit un modèle qui aboutira à un système qui fonctionne correctement, vous êtes libres de modifier les instructions pour les adapter à vous, 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 cheminement.

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

Il y a beaucoup de raisons qui vous pousseraient à vouloir lire ce livre. Une des questions que beaucoup de personnes se posent est « pourquoi se fatiguer à installer manuellement un système Linux depuis le début alors qu'il suffit de télécharger 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 de 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'inclure beaucoup de 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, voire d'autres. Pensez aux CDs, clés USB amorçables et aux systèmes embarqués. Ce sont des champs 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 d'un binaire réparant une faille de sécurité. À moins que vous examiniez vous-mêmes le correctif et que vous l'appliquiez vous-même, vous n'avez aucune garantie que le nouveau paquet du 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 ne pas bénéficier entièrement 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/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 le processeur Power PC. 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 représentant 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/Intel 64 bits.

D'autres éléments doivent être ajoutés ici sur les systèmes 64 bits. Comparé à un système 32 bits, la taille des programmes exécutables est légèrement plus importante et les vitesses d'exécution ne sont pas beaucoup plus rapides. Par exemple, dans le test de la construction de LFS-6.5 sur un système basé sur un processeur bicoeur, nous avons relevé les statistiques suivantes :

```
Temps de construction de l'architecture     Taille de la construction
32 bit       198.5 minutes  648 Mo
64 bit       190.6 minutes  709 Mo
```

Comme vous pouvez le voir, la construction 64 bits n'est plus rapide que de 4% et elle est plus lourde de 9% par rapport à la construction 32 bits. Le gain du passage au système 64 bits est relativement minime. Bien entendu, si vous avez plus de 4 Go de RAM ou si vous voulez manipuler des données qui excèdent 4 Go, les avantages d'un système 64 bits sont substantiels.

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 supporte 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 supporté par LFS car cela interférerait avec l'objectif pédagogique visant à fournir les instructions nécessaires à un simple système Linux de base. Vous pouvez vous référer au projet *Cross Linux From Scratch* pour ce sujet avancé.

Un dernier commentaire sur les systèmes 64 bits. Il y a des paquets qui ne peuvent pour l'instant pas être construits dans un système 64 bits "pur" ou qui impliquent des instructions de construction spécialisées. En général, ces paquets incluent des instructions du langage de l'assembleur spécifiques au 32 bits qui échouent lors de la construction sur un système 64 bits. Ceci inclut certains pilotes de Xorg de *Beyond Linux From Scratch (BLFS)*. On peut contourner la plupart de ces problèmes, mais cela peut impliquer des procédures spécialisées ou des correctifs.

Il faut ajouter un dernier élément concernant les systèmes 64 bits. Il existe de vieux paquets qui ne peuvent pas être actuellement construits dans un système "pur" 64 bits, ou qui exigent des instructions de construction spécifiques. En général, ces paquets contiennent des instructions d'assembleur spécifiques au 32 bits embarquées qui échouent lors d'une construction sur un système 64 bits. Ceci inclut certains pilotes Xorg pour certaines cartes graphiques de base sur *http://xorg.freedesktop.org/releases/individual/driver/*. On peut contourner la plupart de ces problèmes mais cela peut impliquer des procédures ou des correctifs spécifiques.

LFS et les standards

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

- *POSIX.1-2008.*.
- *Filesystem Hierarchy Standard (FHS)*
- *Linux Standard Base (LSB) Specification 4.0*

  La LSB comporte cinq standards séparés : le cœur, C++, le bureau, les langages à l'exécution et l'impression. Outre les exigences génériques, il y a aussi les exigences spécifiques à l'architecture. LFS s'efforce de respecter l'architecture évoquée dans la section précédente.

Remarque

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 LSB

Cœur LSB :	Bash, Binutils, Coreutils, Diffutils, File, Findutils, Gawk, Grep, Gzip, M4, Man-DB, Ncurses, Procps, Psmisc, Sed, Shadow, Tar, Util-linux, Zlib

LSB C++ :	Gcc
LSB bureau :	Aucun
LSB Langage à l'exécution :	Perl
LSB impression :	Aucun
LSB Multiméda :	Aucun

Paquets fournis par BLFS requis pour satisfaire les exigences LSB

Cœur LSB :	At, Batch (partie de At), Bc, Cpio, Ed, Fcrontab, Initd-tools, Lsb_release, PAM, Sendmail (ou Postfix ou Exim)
LSB C++ :	Aucun
LSB bureau :	ATK, Cairo, Desktop-file-utils, Freetype, Fontconfig, Glib2, GTK+2, Icon-naming-utils, Libjpeg, Libpng, Libxml2, MesaLib, Pango, Qt3, Qt4, Xorg
LSB langage à l'exécution :	Python
LSB impression :	CUPS
LSB Multimédia :	Bibliothèques Alsa, NSPR, NSS, OpenSSL, Java, Xdg-utils

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

Cœur LSB :	Aucun
LSB C++ :	Aucun
LSB bureau :	Aucun
LSB Langage à l'exécution :	Aucun
LSB impression :	Aucun
LSB Multimédia :	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.

- 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 des mises à jour des procédures de construction.

- Bash

Ce paquet satisfait une exigence du coeur 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.

- 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 allant au-delà.

- Bison

  Ce paquet contient la version GNU de yacc (*Yet Another Compiler Compiler*, encore un nouveau 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 une base de tests pour d'autres programmes. On ne l'installe que dans l'ensemble d'outils temporaire.

- 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 environnement de travail pour tester d'autres programmes. Il n'est installé que dans la chaîne d'outils temporaires.

- Diffutils

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

- Expect

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

- 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 supportés par Linux.

- File

  Ce paquet contient un outil pour déterminer le type d'un ou plusieurs fichiers donnés. Quelques paquets en ont besoin pour se construire.

- 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 localisation de nombreux paquets.

- Glibc

  Le 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.

- 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 man.

- GRUB

  Ce paquet est le chargeur *Grand Unified Boot*. Ce n'est pas le seul chargeur de démarrage 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.

- 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 possibilités 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 le bas lors de la visualisation d'un fichier. Il est aussi utilisé par Man-DB pour visualiser des pages de man.

- Libtool

  Ce paquet contient le script de support de la bibliothèque générique 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

  Le paquet M4 contient un processeur général de macros textes 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 man. Il a été préféré au paquet man du fait d'une capacité d'internationalisation supérieure. Il fournit le programme man.

- Man-pages

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

- MPC

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

- MPFR

  Le paquet MPFR contient des fonctions pour des maths à précision multiple. Il est exigé par Gcc.

- 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.

- 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 en cours d'exécution. Il est nécessaire pour l'installation et les suites de tests de plusieurs paquets LFS.

- Pkg-config

  Ce paquet fournit un programme pour retourner des métadonnées sur une bibliothèque ou un binaire installé.

- Popt

  Ce paquet est une bibliothèque utilisée par certains programmes pour analyser les entrées en ligne de commande.

- Procps

  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.

- Readline

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

- Sed

  Ce paquet permet d'entrer du texte sans l'ouvrir 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. Il n'est installé que dans la chaîne d'outils temporaire.

- 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.

- Udev

  Ce paquet contient des programmes pour la création dynamique de nœuds de périphériques. C'est une alternative à la création de milliers de périphériques statiques dans le répertoire /dev.

- Util-linux

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

- Vim

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

- 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 XZ ou du format LZMA.

- Zlib

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

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 avoir déjà la capacité d'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 de support 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 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.

- The Linux Users' Guide *http://tldp.org/pub/Linux/docs/ldp-archived/users-guide/*

  Ce guide couvre l'utilisation de différents logiciels Linux. Cette référence est aussi ancienne mais toujours valide.

- Prélecture essentielle *http://lfs.traduc.org/view/astuces/Prelecture.txt*

  C'est une astuce LFS écrite spécifiquement pour les nouveaux utilisateurs Linux. C'est principalement une liste de liens d'excellentes sources d'informations sur une grande gamme de thèmes. Toute personne essayant d'installer LFS devrait au moins avoir une certaine compréhension de la majorité des thèmes de cette astuce.

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. Noter également que certaines distributions placeront les en-tête des logiciels dans un répertoire distinct des paquets, ayant souvent la forme « <nom-du-paquet>-devel » ou « <nom-du-paquet>-dev ». Assurez-vous qu'ils sont installés si votre distribution les fournit.

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

- **Bash-3.2** (/bin/sh devrait être un lien symbolique ou physique vers bash)
- **Binutils-2.17** (les versions supérieure à 2.23.1 ne sont pas recommandées car elles n'ont pas été testées)
- **Bison-2.3** (/usr/bin/yacc devrait être un lien vers bison ou un petit script qui exécute bison)
- **Bzip2-1.0.4**
- **Coreutils-6.9**
- **Diffutils-2.8.1**
- **Findutils-4.2.31**
- **Gawk-3.1.5** (/usr/bin/awk devrait être un lien vers gawk)
- **Gcc-4.1.2** (les versions supérieures à 4.7.2 ne sont pas recommandées car elles n'ont pas été testées)
- **Glibc-2.5.1** (les versions supérieures à 2.17 ne sont pas recommandées car elles n'ont pas été testées)
- **Grep-2.5.1a**
- **Gzip-1.3.12**
- **Noyau Linux 2.6.25** (compilé avec GCC-4.1.2 ou supérieur)

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

  Si le noyau hôte est plus ancien que le 2.6.25, ou s'il n'a pas été compilé avec le compilateur GCC-4.1.2 (ou supérieur), vous devrez remplacer le noyau par un nouveau qui satisfait ces spécifications. Vous pouvez employer deux méthodes pour cela. Vous pouvez d'abord voir si votre distribution Linux fournit un paquet pour le noyau 2.6.25 ou supérieur. Si tel est le cas, vous pouvez l'installer. Si votre distribution n'offre pas un paquet acceptable pour le noyau, ou si vous préférez l'installer, vous pouvez compiler un noyau 2.6 vous-même. Les instructions pour la compilation du noyau et la configuration du chargeur de démarrage (en supposant que le système hôte utilise GRUB) sont au Chapitre 8.
- **M4-1.4.10**
- **Make-3.81**
- **Patch-2.5.4**
- **Perl-5.8.8**
- **Sed-4.1.5**
- **Tar-1.18**
- **Texinfo-4.9**
- **Xz-5.0.0**

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

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

```
cat > version-check.sh << "EOF"
#!/bin/bash
# Simple script to list version numbers of critical development tools

export LC_ALL=C
bash --version | head -n1 | cut -d" " -f2-4
echo "/bin/sh -> `readlink -f /bin/sh`"
echo -n "Binutils: "; ld --version | head -n1 | cut -d" " -f3-
bison --version | head -n1
if [ -e /usr/bin/yacc ];
  then echo "/usr/bin/yacc -> `readlink -f /usr/bin/yacc`";
  else echo "yacc not found"; fi

bzip2 --version 2>&1 < /dev/null | head -n1 | cut -d" " -f1,6-
echo -n "Coreutils: "; chown --version | head -n1 | cut -d")" -f2
diff --version | head -n1
find --version | head -n1
gawk --version | head -n1
if [ -e /usr/bin/awk ];
  then echo "/usr/bin/awk -> `readlink -f /usr/bin/awk`";
  else echo "awk not found"; fi

gcc --version | head -n1
ldd --version | head -n1 | cut -d" " -f2-  # glibc version
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`
sed --version | head -n1
tar --version | head -n1
echo "Texinfo: `makeinfo --version | head -n1`"
xz --version | head -n1

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

bash version-check.sh
```

Typographie

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

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

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

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

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

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

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

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

Mise en évidence

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

http://www.linuxfromscratch.org/

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

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

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

`<TEXTE A 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 man. Pour les instructions d'installation de LFS, ces deux pages man seront situées dans `/usr/share/man/man1/passwd.1`. Quand le

livre utilise `passwd(5)`, il fait spécifiquement référence à `/usr/share/man/man5/passwd.5`. **man passwd** affichera la première page man qu'il trouvera et qui aura une correspondance avec « passwd », à priori `/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 par exemple concernant 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 du système LFS

La troisième 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.

Errata

Le logiciel utilisé pour créer un système LFS est constamment mis à jour et amélioré. Les messages d'avertissements pour la sécurité et les 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 les vulnérabilités en terme de sécurité ou toute autre correction de bogue, merci de visiter *http://www.linuxfromscratch.org/lfs/errata/7.3/* avant de commencer votre construction. Vous devez noter toutes les modifications et les appliquer à la section correspondante du livre pendant votre progression lors de la construction du système LFS.

Partie I. Introduction

Chapitre 1. Introduction

1.1. Comment construire un système LFS

Le système LFS sera construit en utilisant une distribution Linux déjà installée (telle que Debian, Mandrake, Red Hat ou SuSE). 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.

Alternativement à 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 dont vous devez être au courant avant de commencer à travailler sur le Chapitre 5 et les chapitres suivants.

Le Chapitre 5 explique l'installation d'un ensemble de paquets qui formera la suite de développement de base (ou ensemble d'outils) utilisé pour construire le système réel dans le Chapitre 6. Certains de ces paquets sont nécessaires pour résoudre des dépendances circulaires — par exemple, pour compiler un compilateur, vous avez besoin d'un compilateur.

Le Chapitre 5 vous montre aussi comment construire dans une première passe l'ensemble des outils, incluant Binutils et GCC (première passe signifiant basiquement que ces deux paquets principaux seront installés une deuxième fois). La prochaine étape consiste à construire Glibc, la bibliothèque C. Glibc sera compilé par les programmes de l'ensemble d'outils, construits lors de la première passe. Ensuite, une seconde passe de la chaîne d'outils sera lancée. Cette fois, l'ensemble d'outils sera lié dynamiquement avec la Glibc nouvellement construite. Les paquets restants du Chapitre 5 seront construits en utilisant la chaîne d'outils de cette deuxième passe. Lorsque ceci sera fait, le processus d'installation de LFS ne dépendra plus de la distribution hôte, à l'exception du noyau en cours d'exécution.

Cet effort consistant à isoler le nouveau système de la distribution hôte peut sembler excessif. Une explication technique complète est fournie dans Section 5.2, « Notes techniques sur la chaîne d'outils ».

Dans le Chapitre 6, le système LFS complet est construit. Le programme **chroot** (changement de racine) est utilisé pour entrer dans un environnement virtuel et pour lancer un nouveau shell dont le répertoire racine sera initialisé à la partition LFS. Ceci ressemble à redémarrer et donner l'instruction au noyau de monter la partition LFS comme partition racine. Le système ne redémarre pas réellement mais change la racine parce que la création d'un système démarrable (amorçable) réclame un travail supplémentaire qui n'est pas encore nécessaire. L'avantage principal est que se « chrooter » vous permet de continuer à utiliser l'hôte pendant la construction de LFS. En attendant que les compilations d'un paquet se termine, un utilisateur peut passer sur une console virtuelle (VC) différente ou un bureau X et continuer à utiliser son ordinateur comme d'habitude.

Pour terminer l'installation, les scripts de démarrage sont configurés dans le Chapitre 7, le noyau et le chargeur de démarrage sont configurés dans le Chapitre 8. Le Chapitre 9 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 tout ira à sa place, alors que vous vous embarquerez pour l'aventure LFS.

1.2. Quoi de neuf 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 :

-
- Automake 1.13.1
- Binutils 2.23.1
- Bison 2.7
- Check 0.9.9
- Coreutils 8.21
- E2fsprogs 1.42.7
- Gawk 4.0.2
- GCC 4.7.2
- Gettext 0.18.2
- Glibc 2.17
- GMP 5.1.0
- Groff 1.22.1
- IPRoute2 3.8.0
- Kmod 12
- Less 451
- Libpipeline 1.2.2
- Linux 3.8.1
- Man-DB 2.6.3
- Man-pages 3.46
- MPC 1.0.1
- Patch 2.7.1
- Pkg-config 0.27.1
- Psmisc 22.20
- Sed 4.2.2
- TCL 8.6.0
- Texinfo 5.0
- Tzdata 2012j
- Udev 197 (extracted from systemd-197)

- Util-Linux 2.22.2

Ajoutés :

- bash-4.2-fixes-11.patch
- binutils-2.23.1-testsuite_fix-1.patch
- coreutils-8.21-i18n-1.patch
- kbd-1.15.5-backspace-1.patch
- make-3.82-upstream_fixes-3.patch

Supprimés :

- bash-4.2-fixes-8.patch
- binutils-2.22-build_fix-1.patch
- coreutils-8.19-i18n-1.patch
- gcc-4.7.1-startfiles_fix-1.patch
- glibc-2.16.0-fix_test_installation-1.patch
- glibc-2.16.0-res_query_fix-1.patch
- kmod-9-testsuite-1.patch
- make-3.82-upstream_fixes-2.patch
- patch-2.6.1-test_fix-1.patch
- procps-3.2.8-fix_HZ_errors-1.patch
- procps-3.2.8-watch_unicode-1.patch
- sed-4.2.1-testsuite_fixes-1.patch

1.3. Historique des modifications

Il s'agit de la version 7.3 du livre Linux From Scratch, datant du 1 mars 2013. Si ce livre est daté de plus de six mois, une nouvelle et meilleure version est probablement déjà disponible. Pour le savoir, merci de vérifier la présence d'une nouvelle version sur l'un des miroirs via *http://www.linuxfromscratch.org/mirrors.html*.

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

Entrées dans l'historique des modifications:

- 27-02-2013
  - [bdubbs] - Correction d'une erreur potentielle des tests dans procps-ng. Merci à Pierre Labastie pour le correctif. Corrige *#3293*.
- 26-02-2013
  - [bdubbs] - Passage à File-5.13. Corrige *#3292*.
  - [ken] - Nouvelle suppression du programme obsolète resizecons de kbd.
- 24-02-2013

- [matthew] - Passage à IPRoute2-3.8.0. Corrige *#3291*.
- [matthew] - Passage à Linux-3.8. Corrige *#3290*.

- 19-02-2013
  - [bdubbs] - Mise à jour des noms de fichiers pour ajuster les droits des règles du CD-ROM à la section 7.5.1.
- 30-01-2013
  - [bdubbs] - Passage à Coreutils-8.21. Corrige *#3286*.
  - [bdubbs] - Modification des instructions de ncurses pour créer et installer les fichiers .pc.
- 18-02-2013-02-18
  - [bdubbs] - Passage à Texinfo-5.0. Corrige *#3284*.
  - [bdubbs] - Passage à Linux-3.7.9. Corrige *#3281*.
- 29-01-2013
  - [matthew] - Passage à Man-Pages-3.46. Corrige *#3278*.
  - [matthew] - Passage à Linux-3.7.5. Corrige *#3277*.
- 25-01-2013
  - [bdubbs] - Correction des instructions d'installation de procps-ng pour mettre les fichiers aux bons endroits.
- 24-01-2013
  - [bdubbs] - Suppression de bashismes dans le fichier init-functions des scripts de démarrage.
- 24-01-2013
  - [bdubbs] - Passage à e2fsprogs-1.42.7. Corrige *#3274*.
- 22-01-2013-01-22
  - [bdubbs] - Passage à Linux-3.7.4. Corrige *#3273*.
- 21-01-2013-01-21
  - [bdubbs] - Passage à Procps-ng-3.3.6. Corrige *#3095*.
- 20-01-2013-01-20
  - [matthew] - Passage à Linux-3.7.3. Corrige *#3272*.
- 11-01-2013-01-11
  - [bdubbs] - Reformulation des explications sur /etc/modprobe.conf. Corrige *#3270*.
  - [bdubbs] - Mise à jour du script init-net-rules.sh d'udev-lfs pour introduire les périphériques "en*" dans systemd-197.
- 09-01-2013-01-09
  - [bdubbs] - Reformatage des recommandations pour 'Redémarrer le système'.
  - [bdubbs] - Mise à jour des scripts udev-lfs. Passage à systemd-197.
- 02-01-2013
  - [matthew] - Passage à Bash-4.2.42. Corrige *#3268*.
  - [matthew] - Passage à Groff-1.22.1. Corrige *#3266*.

- [matthew] - Passage à Automake-1.13.1. Corrige *#3265*.
- [matthew] - Passage à Coreutils-8.20. Corrige *#3215*.

- 28-12-2012
  - [bdubbs] - On met traceroute dans /bin par cohérence. Corrige *#3264*.
  - [bdubbs] - Correction de l'emplacement du montage de /dev/shm dans chroot. Corrige *#3258*.
  - [matthew] - Déplacement de la construction de Procps avant E2fsprogs car ce dernier a besoin que **ps** soit disponible pendant l'exécution de sa suite de tests.
  - [matthew] - Passage à Gettext-0.18.2. Corrige *#3263*.
  - [matthew] - Passage à Gawk-4.0.2. Corrige *#3262*.
  - [matthew] - Passage à Glibc-2.17. Corrige *#3261*.
  - [matthew] - Passage à Sed-4.2.2. Corrige *#3260*.
  - [matthew] - Passage à GMP-5.1.0. Corrige *#3259*.
  - [matthew] - Passage à Tcl-8.6.0. Corrige *#3257*.
  - [matthew] - Passage à Man-Pages-3.45. Corrige *#3256*.
- 18-12-2012
  - [bdubbs] - Passage à Automake-1.12.6. Corrige *#3253*.
  - [bdubbs] - Passage à Linux-3.7.1. Corrige *#3254*.
- 16-12-2012
  - [matthew] - Passage à Util-Linux-2.22.2. Corrige *#3250*.
  - [matthew] - Suppression de quelques entrées de la liste d'acronymes car ils ne sont plus utilisés dans le livre. Corrige *#3249*. Merci à Chris Staub pour le correctif.
  - [matthew] - Passage à Bison-2.7. Corrige *#3247*.
- 12-12-2012
  - [matthew] - Passage à IPRoute2-3.7.0. Corrige *#3246*.
  - [matthew] - Mise à jour de la liste des programmes installés de Check. Corrige *#3245*. Merci à Chris Staub pour le correctif.
  - [matthew] - Tous les tests de Flex réussissent, donc suppression de l'option "-k" dans l'appel de la suite de tests. Corrige *#3244*.
  - [matthew] - Passage à E2fsprogs-1.42.6. Corrige *#3243*.
  - [matthew] - Suppression d'un paramètre --enable-addons redondant dans le script configure de Glibc. Corrige *#3241*.
  - [matthew] - Mise à jour des informations du serveur IRC. Corrige *#3240*. Merci à Chris Staub pour le correctif.
  - [matthew] - Passage à Kmod-12. Corrige *#3238*.
  - [matthew] - Passage à Linux-3.7. Corrige *#3237*.
- 03-12-2012
  - [bdubbs] - Correction de problèmes de construction dans makefile pour systemd-196/udev-lfs-196.

- 28-11-2012
  - [bdubbs] - Mise à jour du makefile et des instructions pour systemd-196/udev-lfs-196.
- 27-11-2012
  - [matthew] - Passage à Linux-3.6.8. Corrige *#3234*.
- 26-11-2012
  - [bdubbs] - Remise en valeur des exigences de système hôte au chapitre 5 Instructions générales de compilation.
- 22-11-2012
  - [bdubbs] - Passage à systemd-196/udev-lfs-196. Corrige *#3233*.
- 18-11-2012
  - [matthew] - Passage à Linux-3.6.7. Corrige *#3232*.
  - [matthew] - Passage à Automake-1.12.5. Corrige *#3231*.
- 14-11-2012
  - [matthew] - Passage à Tzdata-2012j. Fixes *#3227*.
  - [matthew] - Passage à Binutils-2.23.1. Corrige *#3226*.
- 13-11-2012

  2012-11-14
  - [matthew] - Passage à Tcl-8.5.13. Corrige *#3224*.
  - [matthew] - Passage à Kmod-11. Corrige *#3223*.
  - [matthew] - Passage à Man-Pages-3.44. Corrige *#3222*.
  - [matthew] - Passage à Bison-2.6.5. Corrige *#3221*.
  - [matthew] - Passage à Tzdata-2012i. Corrige *#3220*.
  - [matthew] - Passage à Linux-3.6.6. Corrige *#3219*.
- 03-11-2012
  - [matthew] - Passage à Perl-5.16.2. Corrige *#3218*.
  - [matthew] - Passage à Bash-4.2.39. Corrige *#3217*.
- 02-11-2012
  - [matthew] - Passage des références à Freshmeat.net en Freecode. Merci à Chris Staub pour le signalement et le correctif. Corrige *#3216*.
  - [matthew] - Passage à Check-0.9.9. Corrige *#3214*.
  - [matthew] - Passage à Bison-2.6.4. Corrige *#3212*.
  - [matthew] - Passage à Linux-3.6.5. Corrige *#3211*.
  - [matthew] - Passage à Tzdata-2012h. Corrige *#3209*.
  - [matthew] - Passage à Man-Pages-3.43. Corrige *#3208*.
  - [matthew] - Nettoyage de deux instructions pour Flex, rendu possible par la mise à jour vers Flex-2.5.37. Corrige *3206* et *3210*.

- 01-11-2012
  - [bdubbs] - Passage à systemd/lfs-udev-195. Corrige *#3197*.
- 15-10-2012
  - [bdubbs] - Ajout de remarques aux sections gcc et binutils du chapitre 6 concernant "l'optimisation du temps d'édition des liens" et sur les fichiers supplémentaires construits par gcc. Corrige *#3200*.
  - [bdubbs] - Passage à tzdata-2012f. Corrige *#3205*.
  - [bdubbs] - Mise à jour de la description des programmes installés pour plusieurs paquets. Merci à Chris Staub pour le corrcdsif. Corrige *#3203*.
- 27-09-2012
  - [bdubbs] - Suppression d'utmpdump de sysvinit car il est installé maintenant par util-linux. Merci à Chris Staub pour le correctif. Corrige *#3202*.
  - [bdubbs] - Modification de procps pour ne pas installer le programme kill installé maintenant par util-linux. Corrige *#3201*.
  - [bdubbs] - Passage à linux-3.6.2. Corrige *#3198*.
  - [bdubbs] - Ajout d'instructions de la personnalisation du script boot/shutdown.
- 02-10-2012
  - [matthew] - Passage à IPRoute2-3.6.0. Corrige *#3196*.
  - [matthew] - Passage à Linux-3.6. Corrige *#3195*.
  - [matthew] - Passage à Psmisc-22.20. Corrige *#3194*.
  - [matthew] - Passage à Patch-2.7.1. Corrige *#3193*.
  - [matthew] - Ajout d'un correctif pour permettre à la suite de tests de Kmod de passer sur les systèmes 32 bits. Corrige *#3191*.
  - [matthew] - Passage à GCC-4.7.2. Corrige *#3190*.
  - [matthew] - Utilisation des derniers correctifs en amont pour Make, qui permettent à WebKitGtk de construire avec des processus parallèles. Corrige *#3188*.
  - [matthew] - Passage à Man-DB-2.6.3. Corrige *#3187*.
  - [matthew] - Passage à Libpipeline-1.2.2. Corrige *#3186*.
  - [matthew] - Passage à Automake-1.12.4. Corrige *#3185*.
  - [matthew] - Correction des instructions de Flex ; la création d'un répertoire pour sa documentation est effectuée par le Makefile depuis la r9999.
- 27-09-2012
  - [bdubbs] - Passage à systemd/udev-lfs-193. Corrige *#3192*.
  - [bdubbs] - Passage à systemd/udev-lfs-192. Corrige *#3189*.
- 16-09-2012
  - [matthew] - Passage à Patch-2.7. Corrige *#3182*.
  - [matthew] - Passage à MPC-1.0.1. Corrige *#3181*.
  - [matthew] - Passage à Kmod-10. Corrige *#3180*.

- [matthew] - Correction du lien vers le guide de l'utilisateur Linux. Corrige *#3179*.
- [matthew] - Passage à Less-451. Corrige *#3178*.
- [matthew] - Passage à Bash-4.2.37. Corrige *#3177*.
- [matthew] - Passage à Pkg-Config-0.27.1. Corrige *#3174*.
- [matthew] - Passage à Linux-3.5.4. Corrige *#3173*.

- 05-09-2012
  - [bdubbs] - Ajout de la ligne de ratrapage --action=change à udevadm au script udev pour mieux supporter initramfs.
- 04-09-2012
  - [bdubbs] - Mise à jour de l'exemple des conventions de paquets et de nommage de grub.
  - [bdubbs] - Passage à util-linux-2.22. Corrige *#3145*.
- 02-09-2012
  - [bdubbs] - Ajout d'un correctif à Glibc au chapitre 6 pour corriger le script test-installation.pl plutôt que de l'empêcher seulement de s'exécuter. Corrige *#3175*.
  - [bdubbs] - Passage à systemd-189. Corrige *#3167*.
- 01-09-2012
  - [bdubbs] - Publication de LFS-7.2.

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://www.linuxfromscratch.org/faq/*.

1.4.2. Listes de diffusion

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

Pour connaître les listes disponibles, les conditions d'abonnement, l'emplacement des archives et quelques autres informations, allez sur *http://lfs.traduc.org/support.php*.

1.4.3. IRC

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

1.4.4. Sites miroirs

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

1.4.5. Contacts

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

1.5. Aide

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

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

Nous avons aussi une formidable communauté LFS, volontaire pour offrir une assistance via les listes de discussion et IRC (voir la section Section 1.4, « Ressources » de ce livre). Néanmoins, nous recevons plusieurs questions de support chaque jour et un grand nombre d'entre elles ont une réponse dans la FAQ et dans les listes de discussions. Pour que nous puissions vous offrir la meilleure assistance possible, vous devez faire quelques recherches de votre côté. Ceci nous permet de nous concentrer sur les besoins inhabituels. Si vos recherches ne vous apportent aucune solution, merci d'inclure toutes les informations adéquates (mentionnées ci-dessous) dans votre demande d'assistance.

1.5.1. Éléments à mentionner

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

- La version du livre que vous utilisez (dans ce cas, 7.3)
- La distribution hôte (et sa version) que vous utilisez pour créer LFS
- La sortie de Section vii, « Prérequis du système hôte » [xviii]
- Le paquet ou la section où le problème a été rencontré
- Le message d'erreur exact ou le symptôme reçu
- Notez si vous avez dévié du livre

Remarque

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

1.5.2. Problèmes avec le script configure

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

1.5.3. Problèmes de compilation

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

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

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

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

Cette information n'est pas suffisante pour diagnostiquer correctement le problème car il note seulement que quelque chose s'est mal passé, pas *ce* qui s'est mal passé. La section entière, comme dans l'exemple ci-dessus, est ce qui devrait être sauvée car la commande exécutée et le(s) message(s) d'erreur associé(s) 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. Préparer une nouvelle partition

2.1. Introduction

Dans ce chapitre, on prépare la partition qui contiendra le système LFS. Nous créerons la partition elle-même, lui ajouterons un système de fichiers et nous la monterons.

2.2. Créer une nouvelle partition

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

Un système minimal requiert une partition d'environ 2.8 Gio (giga octets). 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 10 Gio 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 libre suffisant mais temporaire. Compiler des paquets peut demander beaucoup d'espace disque qui sera récupéré après l'installation du paquet.

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

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

Remarque

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

Rappelez-vous de la désignation de la nouvelle partition (par exemple `hda5`). Ce livre y fera référence en tant que la partition LFS. Rappelez-vous aussi de la désignation de la partition `swap`. Ces noms seront nécessaires après pour le fichier `/etc/fstab`.

2.2.1. Autres problématiques du partitionnement

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

2.2.1.1. La partition racine

Une partition racine LFS (à ne pas confondre avec le répertoire `/root`) est de dix giga-octets 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.

2.2.1.2. La partition d'échange

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

L'utilisation de la mémoire d'échange n'est jamais une bonne chose. En général, 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. Votre première réaction lorsque la mémoire d'échange est utilisé devrait être de vérifier si une commande n'est pas déraisonnable, telle que l'essai d'édition d'un fichier de cinq giga-octets. Si l'utilisation de la mémoire d'échange devient un phénomène habituel, la meilleure solution est d'ajouter de la RAM à votre système.

2.2.1.3. 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 – Fort 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 100 méga-octets est parfaitement adaptée.
- /home – Fort recommandée. Partage votre répertoire home et vos paramètres utilisateur entre plusieurs distributions ou constructions de LFS. La taille est en général très importante et dépend de l'espace disque disponible.
- /usr – On utilise généralement une partition /usr séparée si on fournit un serveur pour un client léger ou une station de travail sans disque. Elle n'est normalement pas nécessaire pour LFS. Une taille de cinq giga-octets gèrera la plupart des installations.
- /opt – Ce répertoire est surtout utile pour BLFS où vous pouvez installer plusieurs versions de gros paquets tels que Gnome ou KDE sans mettre les fichiers dans la hiérarchie /usr. Si vous l'utilisez, 5 à 10 giga-octets sont généralement adaptés.
- /tmp – Un répertoire /tmp séparé est rare, mais utile si vous configurez un client léger. Cette partition, si vous l'utilisez, ne nécessitera en général pas plus de deux giga-octets.
- /usr/src – Cette partition est très utile pour fournir un endroit où conserver les fichiers des sources de BLFS et les partager entre des constructions LFS. Vous pouvez aussi l'utiliser comme lieu de construction des paquets BLFS. Une partition raisonnablement grande de 30-50 giga-octets permet d'avoir beaucoup de place.

Vous devez spécifier toute partition que vous voulez voir montée automatiquement au démarrage dans `/etc/fstab`. Les détails sur la façon de spécifier les partitions seront donnés au Section 8.2, « Créer le fichier /etc/fstab ».

2.3. Créer un système de fichiers sur la partition

Maintenant qu'une partition vierge est prête, le système de fichiers peut être créé. Le système le plus communément utilisé dans le monde Linux est le système de fichiers étendu, deuxième version, plus connu sous son acronyme (`ext2`, mais avec les nouveaux disques haute capacité, les systèmes de fichiers journalisés deviennent de plus en

plus populaires. Le système de fichiers étendu, troisième version (ext3) est une amélioration couramment utilisée de ext2, qui ajoute des options de journalisation et qui est compatible avec les utilitaires de E2fsprogs. Nous créerons un système de fichiers ext3. Les instructions de construction d'autres systèmes de fichiers sont disponibles dans *http://lfs.traduc.org/view/blfs-svn/postlfs/filesystems.html*.

Pour créer un système de fichiers ext3 sur la partition LFS, lancez ce qui suit :

```
mke2fs -jv /dev/<xxx>
```

Remplacez <*xxx*> par le nom de la partition LFS (hda5 dans notre exemple précédent).

Remarque

Quelques distributions hôtes utilisent des fonctionnalités personnalisées dans leur outil de création de systèmes de fichiers (e2fsprogs). Ceci peut poser des problèmes lors du démarrage dans votre nouveau LFS au chapitre 9 car toutes ces fonctionnalités ne seront pas supportées par la version d'e2fsprogs installée par LFS ; vous aurez une erreur du type « unsupported filesystem features, upgrade your e2fsprogs ». Pour voir si votre système hôte utilise des améliorations personnalisées, utilisez la commande suivante :

```
debugfs -R feature /dev/<xxx>
```

Si la sortie contient des fonctionnalités autres que has_journal, ext_attr, resize_inode, dir_index, filetype, sparse_super, large_file ou needs_recovery, alors votre système hôte pourrait avoir des améliorations personnalisées. Dans ce cas, pour éviter tout problème ultérieur, vous devez compiler le paquet e2fsprogs et utiliser les binaires résultant de cette compilation pour re-créer le système de fichiers sur votre partition LFS :

```
cd /tmp
tar -xzvf /path/to/sources/e2fsprogs-1.42.7.tar.gz
cd e2fsprogs-1.42.7
mkdir -v build
cd build
../configure
make #note that we intentionally don't 'make install' here!
./misc/mke2fs -jv /dev/<xxx>
cd /tmp
rm -rfv e2fsprogs-1.42.7
```

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

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

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

2.4. Monter la nouvelle partition

Maintenant qu'un système de fichiers a été créé, la partition doit être accessible. Pour cela, la partition a besoin d'être montée sur un point de montage choisi. Pour ce livre, il est supposé que le système de fichiers est monté sous /mnt/lfs, mais le choix du répertoire vous appartient.

Choisissez un point de montage et affectez-le à la variable d'environnement LFS en lançant :

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

Maintenant, créez le point de montage et montez le système de fichiers LFS en lançant :

```
mkdir -pv $LFS
mount -v -t ext3 /dev/<xxx> $LFS
```

Remplacez <*xxx*> par la désignation de la partition LFS.

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

```
mkdir -pv $LFS
mount -v -t ext3 /dev/<xxx> $LFS
mkdir -v $LFS/usr
mount -v -t ext3 /dev/<yyy> $LFS/usr
```

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

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

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

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

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

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

Chapitre 3. Paquets et correctifs

3.1. Introduction

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

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

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

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

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

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

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

Une manière simple de télécharger tous les paquets et les correctifs est d'utiliser *wget-list* comme entrée pour **wget**. Par exemple :

```
wget -i wget-list -P $LFS/sources
```

En outre, à partir de LFS-7.0, il existe un fichier séparé, *md5sums*, qui 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
```

3.2. Tous les paquets

Téléchargez ou obtenez autrement les paquets suivants :

• **Autoconf (2.69) - 1,186 Ko:**
Page d'accueil : *http://www.gnu.org/software/autoconf/*
Téléchargement : *http://ftp.gnu.org/gnu/autoconf/autoconf-2.69.tar.xz*
Somme de contrôle MD5 : `50f97f4159805e374639a73e2636f22e`

• Automake (1.13.1) - 1,392 Ko:
Page d'accueil : *http://www.gnu.org/software/automake/*
Téléchargement : *http://ftp.gnu.org/gnu/automake/automake-1.13.1.tar.xz*
Somme de contrôle MD5 : `a60380ab11e1481376b7747d1b42ced2`

• Bash (4.2) - 6,845 Ko:
Page d'accueil : *http://www.gnu.org/software/bash/*
Téléchargement : *http://ftp.gnu.org/gnu/bash/bash-4.2.tar.gz*
Somme de contrôle MD5 : `3fb927c7c33022f1c327f14a81c0d4b0`

• Binutils (2.23.1) - 20,953 Ko:
Page d'accueil : *http://www.gnu.org/software/binutils/*
Téléchargement : *http://ftp.gnu.org/gnu/binutils/binutils-2.23.1.tar.bz2*
Somme de contrôle MD5 : `33adb18c3048d057ac58d07a3f1adb38`

• Bison (2.7) - 1,735 Ko:
Page d'accueil : *http://www.gnu.org/software/bison/*
Téléchargement : *http://ftp.gnu.org/gnu/bison/bison-2.7.tar.xz*
Somme de contrôle MD5 : `234cdfac99257cf99ac4a03c898f37b9`

• Bzip2 (1.0.6) - 764 Ko:
Page d'accueil : *http://www.bzip.org/*
Téléchargement : *http://www.bzip.org/1.0.6/bzip2-1.0.6.tar.gz*
Somme de contrôle MD5 : `00b516f4704d4a7cb50a1d97e6e8e15b`

• Check (0.9.9) - 589 Ko :
Page d'accueil : *http://check.sourceforge.net/*
Téléchargement : *http://sourceforge.net/projects/check/files/check/0.9.9/check-0.9.9.tar.gz*
Somme de contrôle MD5 : `f3702f2fcfc19ce3f62dca66c241a168`

• Coreutils (8.21) - 5,248 Ko:
Page d'accueil : *http://www.gnu.org/software/coreutils/*
Téléchargement : *http://ftp.gnu.org/gnu/coreutils/coreutils-8.21.tar.xz*
Somme de contrôle MD5 : `065ba41828644eca5dd8163446de5d64`

• DejaGNU (1.5) - 563 Ko:
Page d'accueil : *http://www.gnu.org/software/dejagnu/*
Téléchargement : *http://ftp.gnu.org/gnu/dejagnu/dejagnu-1.5.tar.gz*
Somme de contrôle MD5 : `3df1cbca885e751e22d3ebd1ac64dc3c`

• Diffutils (3.2) - 1,976 Ko:
Page d'accueil : *http://www.gnu.org/software/diffutils/*
Téléchargement : *http://ftp.gnu.org/gnu/diffutils/diffutils-3.2.tar.gz*
Somme de contrôle MD5 : `22e4deef5d8949a727b159d6bc65c1cc`

• E2fsprogs (1.42.7) - 5,856 Ko:
Page d'accueil : *http://e2fsprogs.sourceforge.net/*
Téléchargement : *http://prdownloads.sourceforge.net/e2fsprogs/e2fsprogs-1.42.7.tar.gz*
Somme de contrôle MD5 : `a1ec22ef003688dae9f76c74881b22b9`

• Expect (5.45) - 614 Ko:
Page d'accueil : *http://expect.sourceforge.net/*
Téléchargement : *http://prdownloads.sourceforge.net/expect/expect5.45.tar.gz*
Somme de contrôle MD5 : `44e1a4f4c877e9ddc5a542dfa7ecc92b`

• File (5.13) - 627 Ko:
Page d'accueil : *http://www.darwinsys.com/file/*
Téléchargement : *ftp://ftp.astron.com/pub/file/file-5.13.tar.gz*
Somme de contrôle MD5 : `d60c1364ba956eff7d21f8250808fc6d`

Remarque

Il se peut que le fichier (5.13) ne soit plus disponible à l'emplacement indiqué. Les administrateurs du site de l'emplacement principal de téléchargement suppriment régulièrement les anciennes versions lorsque de nouvelles sortent. Vous pouvez trouver un autre emplacement pour le téléchargement qui peut conserver la bonne version disponible sur *http://www.linuxfromscratch.org/lfs/download.html#ftp*.

• Findutils (4.4.2) - 2,100 Ko:
Page d'accueil : *http://www.gnu.org/software/findutils/*
Téléchargement : *http://ftp.gnu.org/gnu/findutils/findutils-4.4.2.tar.gz*
Somme de contrôle MD5 : `351cc4adb07d54877fa15f75fb77d39f`

• Flex (2.5.37) - 1,280 Ko:
Page d'accueil : *http://flex.sourceforge.net*
Téléchargement : *http://prdownloads.sourceforge.net/flex/flex-2.5.37.tar.bz2*
Somme de contrôle MD5 : `c75940e1fc25108f2a7b3ef42abdae06`

• Gawk (4.0.2) - 1,589 Ko:
Page d'accueil : *http://www.gnu.org/software/gawk/*
Téléchargement : *http://ftp.gnu.org/gnu/gawk/gawk-4.0.2.tar.xz*
Somme de contrôle MD5 : `8a9b2f1170ac9dcd3eb13716b5ec4021`

• GCC (4.7.2) - 80,946 Ko:
Page d'accueil : *http://gcc.gnu.org/*
Téléchargement : *http://ftp.gnu.org/gnu/gcc/gcc-4.7.2/gcc-4.7.2.tar.bz2*
Somme de contrôle MD5 : `cc308a0891e778cfda7a151ab8a6e762`

• GDBM (1.10) - 640 Ko :
Page d'accueil : *http://www.gnu.org/software/gdbm/*
Téléchargement : *http://ftp.gnu.org/gnu/gdbm/gdbm-1.10.tar.gz*
Somme de contrôle MD5 : `88770493c2559dc80b561293e39d3570`

• Gettext (0.18.2) - 15,330 Ko:
Page d'accueil : *http://www.gnu.org/software/gettext/*
Téléchargement : *http://ftp.gnu.org/gnu/gettext/gettext-0.18.2.tar.gz*
Somme de contrôle MD5 : `0c86e5af70c195ab8bd651d17d783928`

• Glibc (2.17) - 10,725 Ko :
Page d'accueil : *http://www.gnu.org/software/libc/*
Téléchargement : *http://ftp.gnu.org/gnu/glibc/glibc-2.17.tar.xz*
Somme de contrôle MD5 : `87bf675c8ee523ebda4803e8e1cec638`

• GMP (5.1.0) - 1,764 Ko:
Page d'accueil : *http://www.gnu.org/software/gmp/*
Téléchargement : *ftp://ftp.gmplib.org/pub/gmp-5.1.0/gmp-5.1.0.tar.xz*
Somme de contrôle MD5 : `69fd511e435f81a0334f14f00c7d77b3`

• Grep (2.14) - 1,172 Ko:
Page d'accueil : *http://www.gnu.org/software/grep/*
Téléchargement : *http://ftp.gnu.org/gnu/grep/grep-2.14.tar.xz*
Somme de contrôle MD5 : `d4a3f03849d1e17ce56ab76aa5a24cab`

• Groff (1.22.1) - 3,920 Ko:
Page d'accueil : *http://www.gnu.org/software/groff/*
Téléchargement : *http://ftp.gnu.org/gnu/groff/groff-1.22.1.tar.gz*
Somme de contrôle MD5 : `875c9c628b8b78a0c325000a43ebb964`

• GRUB (2.00) - 5,016 Ko:
Page d'accueil : *http://www.gnu.org/software/grub/*
Téléchargement : *http://ftp.gnu.org/gnu/grub/grub-2.00.tar.xz*
Somme de contrôle MD5 : `a1043102fbc7bcedbf53e7ee3d17ab91`

• Gzip (1.5) - 704 Ko:
Page d'accueil : *http://www.gnu.org/software/gzip/*
Téléchargement : *http://ftp.gnu.org/gnu/gzip/gzip-1.5.tar.xz*
Somme de contrôle MD5 : `2a431e169b6f62f7332ef6d47cc53bae`

• Iana-Etc (2.30) - 201 Ko:
Page d'accueil : *http://freshmeat.net/projects/iana-etc/*
Téléchargement : *http://anduin.linuxfromscratch.org/sources/LFS/lfs-packages/conglomeration//iana-etc/iana-etc-2.30.tar.bz2*
Somme de contrôle MD5 : `3ba3afb1d1b261383d247f46cb135ee8`

• Inetutils (1.9.1) - 1,941 Ko:
Page d'accueil : *http://www.gnu.org/software/inetutils/*
Téléchargement : *http://ftp.gnu.org/gnu/inetutils/inetutils-1.9.1.tar.gz*
Somme de contrôle MD5 : `944f7196a2b3dba2d400e9088576000c`

• IPRoute2 (3.8.0) - 398 Ko:
Page d'accueil : *http://www.kernel.org/pub/linux/utils/net/iproute2/*
Téléchargement : *http://www.kernel.org/pub/linux/utils/net/iproute2/iproute2-3.8.0.tar.xz*
Somme de contrôle MD5 : `951622fd770428116dc165acba375414`

• Kbd (1.15.5) - 1,621 Ko:
Page d'accueil : *http://ftp.altlinux.org/pub/people/legion/kbd*
Téléchargement : *http://ftp.altlinux.org/pub/people/legion/kbd/kbd-1.15.5.tar.gz*
Somme de contrôle MD5 : `8143e179a0f3c25646ce5085e8777200`

• Kmod (12) - 1,245 Ko:
Téléchargement : *http://www.kernel.org/pub/linux/utils/kernel/kmod/kmod-12.tar.xz*
Somme de contrôle MD5 : `3d63b146c8ee5a04dfbef4be97f8226b`

• Less (451) - 303 Ko:
Page d'accueil : *http://www.greenwoodsoftware.com/less/*
Téléchargement : *http://www.greenwoodsoftware.com/less/less-451.tar.gz*
Somme de contrôle MD5 : `765f082658002b2b46b86af4a0da1842`

• LFS-Bootscripts (20130123) - 33 Ko:
Téléchargement : *http://www.linuxfromscratch.org/lfs/downloads/7.3/lfs-bootscripts-20130123.tar.bz2*
Somme de contrôle MD5 : `2a53fcba68e9f5ed6770c47f05987959`

• Libpipeline (1.2.2) - 733 Ko :
Page d'accueil : *http://libpipeline.nongnu.org/*
Téléchargement : *http://download.savannah.gnu.org/releases/libpipeline/libpipeline-1.2.2.tar.gz*
Somme de contrôle MD5 : `4367a3f598d171fd43dfa8620ed16d55`

• Libtool (2.4.2) - 2,571 Ko:
Page d'accueil : *http://www.gnu.org/software/libtool/*
Téléchargement : *http://ftp.gnu.org/gnu/libtool/libtool-2.4.2.tar.gz*
Somme de contrôle MD5 : `d2f3b7d4627e69e13514a40e72a24d50`

• Linux (3.8.1) - 69,329 Ko:
Page d'accueil : *http://www.kernel.org/*
Téléchargement : *http://www.kernel.org/pub/linux/kernel/v3.x/linux-3.8.1.tar.xz*
Somme de contrôle MD5 : `093b172f907d5455a6746418ad18f5bc`

Remarque

Le noyau Linux est régulièrement mis à jour, souvent suite à la découverte de failles de sécurité. Vous devriez utiliser la version 3.8.x la plus récente disponible du noyau, sauf si la page d'errata dit autre chose.

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 (1.4.16) - 1,229 Ko:
Page d'accueil : *http://www.gnu.org/software/m4/*
Téléchargement : *http://ftp.gnu.org/gnu/m4/m4-1.4.16.tar.bz2*
Somme de contrôle MD5 : `8a7cef47fecab6272eb86a6be6363b2f`

• Make (3.82) - 1,213 Ko:
Page d'accueil : *http://www.gnu.org/software/make/*
Téléchargement : *http://ftp.gnu.org/gnu/make/make-3.82.tar.bz2*
Somme de contrôle MD5 : `1a11100f3c63fcf5753818e59d63088f`

• Man-DB (2.6.3) - 1,356 Ko:
Page d'accueil : *http://www.nongnu.org/man-db/*
Téléchargement : *http://download.savannah.gnu.org/releases/man-db/man-db-2.6.3.tar.xz*
Somme de contrôle MD5 : `a593a095599ae97bcacf8d038659a146`

• Man-pages (3.46) - 1,092 Ko:
Page d'accueil : *http://www.kernel.org/doc/man-pages/*
Téléchargement : *http://www.kernel.org/pub/linux/docs/man-pages/man-pages-3.46.tar.xz*
Somme de contrôle MD5 : `eba99dc469e526c3de2e91fe0a0ddc43`

• MPFR (3.1.1) - 1,047 Ko:
Page d'accueil : *http://www.mpfr.org/*
Téléchargement : *http://www.mpfr.org/mpfr-3.1.1/mpfr-3.1.1.tar.xz*
Somme de contrôle MD5 : `91d51c41fcf2799e4ee7a7126fc95c17`

• MPC (1.0.1) - 610 Ko :
Page d'accueil : *http://www.multiprecision.org/*
Téléchargement : *http://www.multiprecision.org/mpc/download/mpc-1.0.1.tar.gz*
Somme de contrôle MD5 : `b32a2e1a3daa392372fbd586d1ed3679`

• Ncurses (5.9) - 2,760 Ko:
Page d'accueil : *http://www.gnu.org/software/ncurses/*
Téléchargement : *ftp://ftp.gnu.org/gnu/ncurses/ncurses-5.9.tar.gz*
Somme de contrôle MD5 : `8cb9c412e5f2d96bc6f459aa8c6282a1`

• Patch (2.7.1) - 660 Ko:
Page d'accueil : *http://savannah.gnu.org/projects/patch/*
Téléchargement : *http://ftp.gnu.org/gnu/patch/patch-2.7.1.tar.xz*
Somme de contrôle MD5 : `e9ae5393426d3ad783a300a338c09b72`

• Perl (5.16.2) - 13,403 Ko:
Page d'accueil : *http://www.perl.org/*
Téléchargement : *http://www.cpan.org/src/5.0/perl-5.16.2.tar.bz2*
Somme de contrôle MD5 : `2818ab01672f005a4e552a713aa27b08`

• Pkg-config (0.27.1) - 1,864 Ko :
Page d'accueil : *http://www.freedesktop.org/wiki/Software/pkg-config*
Téléchargement : *http://pkgconfig.freedesktop.org/releases/pkg-config-0.27.1.tar.gz*
Somme de contrôle MD5 : `5392b4e3372879c5bf856173b418d6a2`

• Procps (3.3.6) - 528 Ko:
Page d'accueil : *http://sourceforge.net/projects/procps-ng*
Téléchargement : *http://sourceforge.net/projects/procps-ng/files/Production/procps-ng-3.3.6.tar.xz*
Somme de contrôle MD5 : `0a050d9be531921db3cd38f1371e73e3`

• Psmisc (22.20) - 422 Ko:
Page d'accueil : *http://psmisc.sourceforge.net/*
Téléchargement : *http://prdownloads.sourceforge.net/psmisc/psmisc-22.20.tar.gz*
Somme de contrôle MD5 : `a25fc99a6dc7fa7ae6e4549be80b401f`

• Readline (6.2) - 2,225 Ko:
Page d'accueil : *http://cnswww.cns.cwru.edu/php/chet/readline/rltop.html*
Téléchargement : *http://ftp.gnu.org/gnu/readline/readline-6.2.tar.gz*
Somme de contrôle MD5 : `67948acb2ca081f23359d0256e9a271c`

• Sed (4.2.2) - 1,035 Ko:
Page d'accueil : *http://www.gnu.org/software/sed/*
Téléchargement : *http://ftp.gnu.org/gnu/sed/sed-4.2.2.tar.bz2*
Somme de contrôle MD5 : `7ffe1c7cdc3233e1e0c4b502df253974`

• Shadow (4.1.5.1) - 2,142 Ko:
Page d'accueil : *http://pkg-shadow.alioth.debian.org/*
Téléchargement : *http://pkg-shadow.alioth.debian.org/releases/shadow-4.1.5.1.tar.bz2*
Somme de contrôle MD5 : `a00449aa439c69287b6d472191dc2247`

• Sysklogd (1.5) - 85 Ko:
Page d'accueil : *http://www.infodrom.org/projects/sysklogd/*
Téléchargement : *http://www.infodrom.org/projects/sysklogd/download/sysklogd-1.5.tar.gz*
Somme de contrôle MD5 : `e053094e8103165f98ddafe828f6ae4b`

• Sysvinit (2.88dsf) - 108 Ko:
Page d'accueil : *http://savannah.nongnu.org/projects/sysvinit*
Téléchargement : *http://download.savannah.gnu.org/releases/sysvinit/sysvinit-2.88dsf.tar.bz2*
Somme de contrôle MD5 : `6eda8a97b86e0a6f59dabbf25202aa6f`

• Tar (1.26) - 2,285 Ko:
Page d'accueil : *http://www.gnu.org/software/tar/*
Téléchargement : *http://ftp.gnu.org/gnu/tar/tar-1.26.tar.bz2*
Somme de contrôle MD5 : `2cee42a2ff4f1cd4f9298eeeb2264519`

• Tcl (8.6.0) - 8,435 Ko:
Page d'accueil : *http://tcl.sourceforge.net/*
Téléchargement : *http://prdownloads.sourceforge.net/tcl/tcl8.6.0-src.tar.gz*
Somme de contrôle MD5 : `573aa5fe678e9185ef2b3c56b24658d3`

• Time Zone Data (2012j) - 209 Ko :
Page d'accueil : *http://www.iana.org/time-zones*
Téléchargement : *http://www.iana.org//time-zones/repository/releases/tzdata2012j.tar.gz*
Somme de contrôle MD5 : `ba2f92ae7ad099090e8f86cff2f2d799`

• Texinfo (5.0) - 3,000 Ko:
Page d'accueil : *http://www.gnu.org/software/texinfo/*
Téléchargement : *http://ftp.gnu.org/gnu/texinfo/texinfo-5.0.tar.xz*
Somme de contrôle MD5 : `ef2fad34c71ddc95b20c7d6a08c0d7a6`

• Systemd (197) - 2,012 Ko :
Page d'accueil : *http://www.freedesktop.org/wiki/Software/systemd/*
Téléchargement : *http://www.freedesktop.org/software/systemd/systemd-197.tar.xz*
Somme de contrôle MD5 : `56a860dceadfafe59f40141eb5223743`

• Udev-lfs, archive tar - 17 Ko:
Téléchargement : *http://anduin.linuxfromscratch.org/sources/other/udev-lfs-197-2.tar.bz2*
Somme de contrôle MD5 : `f4272c121514caf0c2a6245fbffeb047`

• Util-linux (2.22.2) - 3,028 Ko:
Page d'accueil : *http://userweb.kernel.org/~kzak/util-linux/*
Téléchargement : *http://www.kernel.org/pub/linux/utils/util-linux/v2.22/util-linux-2.22.2.tar.xz*
Somme de contrôle MD5 : `eeacbfdd2556acd899a2d0ffdb446185`

• Vim (7.3) - 8,675 Ko:
Page d'accueil : *http://www.vim.org*
Téléchargement : *ftp://ftp.vim.org/pub/vim/unix/vim-7.3.tar.bz2*
Somme de contrôle MD5 : `5b9510a17074e2b37d8bb38ae09edbf2`

• Xz Utils (5.0.4) - 894 Ko:
Page d'accueil : *http://tukaani.org/xz*
Téléchargement : *http://tukaani.org/xz/xz-5.0.4.tar.xz*
Somme de contrôle MD5 : `161015c4a65b1f293d31810e1df93090`

• Zlib (1.2.7) - 493 Ko:
Page d'accueil : *http://www.zlib.net/*
Téléchargement : *http://www.zlib.net/zlib-1.2.7.tar.bz2*
Somme de contrôle MD5 : `2ab442d169156f34c379c968f3f482dd`

Taille totale de ces paquets : environ 305 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 :

• Bash Correctifs d'origine - 55 Ko :
Téléchargement : *http://www.linuxfromscratch.org/patches/lfs/7.3/bash-4.2-fixes-11.patch*
Somme de contrôle MD5 : `366672f68e7cf214bdbef2ef14b13c23`

• Binutils correctif suite de tests - 2.0 Ko :
Téléchargement : *http://www.linuxfromscratch.org/patches/lfs/7.3/binutils-2.23.1-testsuite_fix-1.patch*
Somme de contrôle MD5 : `cb47fae1bc572d45f4b0cff8ae8ecba8`

• Bzip2 Correctif documentation - 1.6 Ko :
Téléchargement : *http://www.linuxfromscratch.org/patches/lfs/7.3/bzip2-1.0.6-install_docs-1.patch*
Somme de contrôle MD5 : `6a5ac7e89b791aae556de0f745916f7f`

• Coreutils Correctif pour l'internationalisation - 132 Ko:
Téléchargement : *http://www.linuxfromscratch.org/patches/lfs/7.3/coreutils-8.21-i18n-1.patch*
Somme de contrôle MD5 : `ada0ea6e1c00c4b7e0d634f49827943e`

• Flex correctif Tests de régression - 2.8 Ko :
Téléchargement : *http://www.linuxfromscratch.org/patches/lfs/7.3/flex-2.5.37-bison-2.6.1-1.patch*
Somme de contrôle MD5 : `d5b001ef9bdbbe32e2f27576d97d8ff0`

• Kbd Correctif réparant Backspace/Delete - 12 Ko:
Téléchargement : *http://www.linuxfromscratch.org/patches/lfs/7.3/kbd-1.15.5-backspace-1.patch*
Somme de contrôle MD5 : `f75cca16a38da6caa7d52151f7136895`

• Make Correctifs en amont - 10 Ko :
Téléchargement : *http://www.linuxfromscratch.org/patches/lfs/7.3/make-3.82-upstream_fixes-3.patch*
Somme de contrôle MD5 : `95027ab5b53d01699845d9b7e1dc878d`

• Perl correctif Libc - 1.6 Ko :
Téléchargement : *http://www.linuxfromscratch.org/patches/lfs/7.3/perl-5.16.2-libc-1.patch*
Somme de contrôle MD5 : `daf5c64fd7311e9249668426805 35f8f`

• Readline correctifs d'origine - 1.3 Ko :
Téléchargement : *http://www.linuxfromscratch.org/patches/lfs/7.3/readline-6.2-fixes-1.patch*
Somme de contrôle MD5 : `3c185f7b76001d3d0af614f6f2cd5dfa`

Taille totale de ces correctifs : environ 218.3 Ko

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

Chapitre 4. Dernières préparations

4.1. À propos de $LFS

Tout au long de ce livre, la variable d'environnement LFS sera utilisée. Il est vital que cette variable soit toujours définie. Elle doit pointer vers le point de montage choisi pour la partition LFS. Vérifiez que votre variable LFS est correctement configurée avec :

```
echo $LFS
```

Assurez-vous que la sortie affiche le chemin vers le point de montage de la partition LFS, c'est-à-dire /mnt/lfs si vous avez suivi l'exemple fourni. Si cet affichage est mauvais, vous pouvez toujours initialiser la variable avec :

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

Avoir cette variable initialisée est tout à votre bénéfice car des commandes telles que **mkdir $LFS/tools** peuvent être saisies de façon littérale. Votre shell remplacera « $LFS » par « /mnt/lfs » (ou par ce chemin avec lequel vous avez initialisé la variable) lorsqu'il exécutera la ligne de commande.

N'oubliez pas de vérifier que $LFS est initialisé à chaque fois que vous entrez dans l'environnement (par exemple, avec **su** pour root ou un autre utilisateur).

4.2. Créer le répertoire $LFS/tools

Tous les programmes compilés dans le Chapitre 5 seront installés dans $LFS/tools pour les tenir séparés des programmes compilés dans le Chapitre 6. Les programmes compilés ici sont seulement des outils temporaires et ne prendront pas part au système LFS final. En les conservant dans un répertoire séparé, nous pourrons facilement les supprimer plus tard. Ceci nous aide aussi à les empêcher de finir dans les répertoires de production de votre hôte (facile à faire par accident dans le Chapitre 5).

Créez le répertoire requis en lançant la commande suivante en tant qu'utilisateur root :

```
mkdir -v $LFS/tools
```

La prochaine étape consiste en la création du lien symbolique /tools sur votre système hôte. Il pointera vers le répertoire que vous venez de créer sur la partition LFS. Lancez cette commande en tant qu'utilisateur root :

```
ln -sv $LFS/tools /
```

Remarque

La commande ci-dessus est correcte. La commande **ln** a quelques variations syntaxiques, assurez-vous de vérifier **info coreutils ln** et ln(1) avant de signaler ce que vous pensez être une erreur.

Le lien symbolique créé nous permet de compiler notre ensemble d'outils de façon à ce qu'il se réfère à /tools, ce qui signifie que le compilateur, l'assembleur et l'éditeur de liens fonctionneront tous dans le chapitre 5 (alors que nous utilisons toujours quelques outils provenant de l'hôte) et dans le suivant (lorsque nous serons en « chrooted » sur la partition LFS).

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, nous recommandons de construire les paquets dans ce chapitre 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 `lfs`) utilisez-le 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 en 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 le groupe et 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 à `$LFS/tools` en indiquant que `lfs` est le propriétaire du répertoire :

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

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

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

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

```
su - lfs
```

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

4.4. Configurer l'environnement

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

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

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

La nouvelle instance du shell est un shell *non-login*, qui ne lit donc pas les fichiers `/etc/profile` ou `.bash_profile`, mais plutôt le fichier `.bashrc` file. 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=/tools/bin:/bin:/usr/bin
export LFS LC_ALL LFS_TGT PATH
EOF
```

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

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 les modes par défaut sont utilisés par l'appel système `open(2)` les nouveaux fichiers finiront avec les droits 644 et les répertoires avec ceux 755).

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

La variable `LC_ALL` contrôle la localisation de certains programmes, faisant que leurs messages suivent les conventions d'un pays spécifié. Si le système hôte utilise une version de Glibc plus ancienne que la 2.2.4, avoir `LC_ALL` initialisé à quelque chose d'autre que « POSIX » ou « C » (pendant ce chapitre) pourrait poser des problèmes si vous quittez l'environnement chroot et souhaitez y retourner plus tard. Initialiser `LC_ALL` à « POSIX » ou « C » (les deux sont équivalents) nous assure que tout fonctionnera comme attendu dans l'environnement chroot.

La variable LFS_TGT initialise une description de machine compatible mais par défaut lors de la construction de notre compilateur et de notre éditeur de liens croisés et lors de la compilation de notre chaîne d'outils temporaire. Vous trouverez plus d'informations dans Section 5.2, « Notes techniques sur la chaîne d'outils ».

En plaçant /tools/bin au début du PATH standard, tous les programmes installés dans Chapitre 5 sont récupérés par le shell immédiatement après leur installation. Ceci, combiné avec la désactivation du hachage, limite le risque que d'anciens programmes de l'hôte soient utilisés alors que les mêmes programmes sont disponibles depuis l'environnement du chapitre 5.

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

```
source ~/.bash_profile
```

4.5. À propos des SBU

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

La mesure 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 « 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 *45 minutes* pour construire ce paquet. Heureusement, la plupart des temps de construction sont bien plus courts que celui de Binutils.

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

Si vous souhaitez voir des chronométrages réels pour des machines spécifiques, nous recommandons la page d'accueil de *http://www.linuxfromscratch.org/~sbu/*.

Remarque

Pour beaucoup de systèmes modernes avec plusieurs processeurs (ou coeurs), 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 bicoeur peut supporter deux processus simultanés avec :

```
export MAKEFLAGS='-j 2'
```

ou simplement en construisant avec :

```
make -j2
```

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

4.6. À propos des suites de tests

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

Certaines des suites de tests sont plus importantes que d'autres. Par exemple, les suites de tests des paquets formant le cœur de l'ensemble des outils—GCC, Binutils, and 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 se terminer, surtout sur du matériel lent, mais ils sont fortement recommandés

Remarque

L'expérience nous a montré qu'il y a peu à gagner en lançant ces suites de tests au Chapitre 5. Il n'y a pas d'échappatoire au fait que le système hôte exerce toujours une influence sur les tests dans ce chapitre, occasionnant fréquemment des échecs étonnants et inexplicables. Comme les outils construits dans le Chapitre 5 sont temporaires et éventuellement supprimés, pour le lecteur habituel de ce livre, nous recommandons de ne pas lancer les suites de tests dans le Chapitre 5 pour l'utilisateur de base. Les instructions de lancement de ces suites de test sont fournies pour les testeurs et les développeurs mais elles sont réellement optionnelles pour tous les autres.

Un problème commun lors du lancement des suites de test pour Binutils et GCC est de manquer de pseudo-terminaux (PTY). Le symptôme est un nombre inhabituellement élevé de tests ayant échoué. Ceci peut arriver pour un certain nombre de raisons. La plus raisonnable est que le système hôte ne dispose pas du système de fichiers `devpts` configuré correctement. Ce problème est traité avec beaucoup plus de détails sur *http://www.linuxfromscratch.org//lfs/faq.html#no-ptys*.

Quelquefois, les suites de test des paquets échoueront mais pour des raisons dont les développeurs sont conscients et qu'ils ont estimées non critique. Consultez les traces sur *http://www.linuxfromscratch.org/lfs/build-logs/7.3/* pour vérifier si ces échecs sont attendus. Ce site est valide pour tous les tests effectués dans ce livre.

Chapitre 5. Construire un système temporaire

5.1. Introduction

Ce chapitre montre comment construire un système Linux minimal. Ce système ne contiendra que les outils nécessaires pour commencer la construction du système LFS final au Chapitre 6 et créer un environnement de travail avec plus de facilité pour l'utilisateur que ne le permettrait un environnement minimum.

Il y a deux étapes dans la construction de ce système minimal. La première étape consiste à construire une chaîne d'outils tout nouveau et indépendant de l'hôte (compilateur, assembleur, éditeur de liens, bibliothèques et quelques outils). La deuxième étape utilise cet chaîne d'outils pour construire tous les autres outils essentiels.

Les fichiers compilés dans ce chapitre vont être installés sous le répertoire `$LFS/tools` de façon à les garder séparés des fichiers installés dans le chapitre suivant et des répertoires de production de votre hôte. Comme tous les paquets compilés ici sont simplement temporaires, nous ne voulons pas polluer le futur système LFS.

5.2. 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 du Chapitre 5 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.

Remarque

Avant de continuer, faites attention au nom de la plateforme de travail, souvent appelée la triplette cible. Une façon simple de déterminer le nom de la triplette cible est de lancer le script **config.guess** venant avec le source pour 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*.

De même, faites attention au nom de l'éditeur de liens de la plateforme, souvent appelé le chargeur dynamique (à ne pas confondre avec l'éditeur de liens **ld** faisant partie de Binutils). Le chargeur dynamique fourni par Glibc trouve et charge les bibliothèques partagées nécessaires à un programme pour s'exécuter, puis l'exécute. Le nom de l'éditeur dynamique pour une machine Intel 32 bits sera `ld-linux.so.2`. 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.

Quelques points techniques sur la façon dont fonctionne la méthode de construction du Chapitre 5 :

- Un léger ajustement du nom de la plateforme de travail, en modifiant le champ "vendor" de la triplette cible via la variable `LFS_TGT`, assure que la première construction de Binutils et de GCC produira un éditeur de liens et un compilateur croisés compatibles. Au lieu de produire des binaires pour une autre architecture, l'éditeur de liens et le compilateur croisé vont produire des binaires compatibles avec le matériel actuel.
- Les bibliothèques temporaires sont compilées de manière croisée. Puisqu'un compilateur croisé, par nature, ne peut pas se baser sur quoique ce soit issu de son système hôte, cette méthode supprime toute possibilité de contamination du système cible en diminuant les chances des en-têtes ou des bibliothèques du système hôte d'être incluses dans les nouveaux outils. La compilation croisée offre aussi la possibilité de construire à la fois des bibliothèques 32 et 64 bits sur du matériel gérant le 64 bits.
- Une manipulation attentionnée des sources de GCC dira au compilateur l'éditeur de liens dynamiques cible à utiliser.

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 cassé, 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, `/tools/bin` et `/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 commutateur *`--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 commutateur *`--verbose`* à l'éditeur de liens. Par exemple, **`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. Un exemple de ce qui peut être vu pendant son exécution de **configure** est :

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

C'est important pour les raisons mentionnées ci-dessus. Cela démontre aussi que le script configure de GCC ne cherche pas les répertoires 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 : **`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 tests. Par exemple, **`gcc -v dummy.c`** affichera des informations détaillées sur les étapes du préprocesseur, de la compilation et de l'assemblage ceci comprenant les chemins de recherche inclus par **gcc** et leur ordre.

Ce qui est ensuite installé est 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 suivant installé 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, **i686-lfs-linux-gnu-gcc**. Les outils binaires et les en-têtes du noyau peuvent être un peu plus compliqués. Du coup, ne prenez pas de risque et utilisez les options disponibles de configure pour renforcer 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="i686-lfs-gnu-gcc"`* pour contrôler les outils binaires

utilisés, et l'utilisation des commutateurs *-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 terme de machinerie de construction et ne repose généralement pas sur la chaîne d'outils par défaut.

Lors de la seconde passe de Binutils, nous sommes capables d'utiliser l'option *--with-lib-path* de configure pour contrôler le chemin de recherche des bibliothèques de **ld**.

Pour la deuxième passe de GCC, ses sources doivent aussi être modifiées pour dire à GCC d'utiliser le nouvel éditeur de liens dynamiques. Un échec pour faire cela aura pour conséquence que les GCC eux-mêmes auront le même nom que l'éditeur de liens dynamique du répertoire `/lib` du système hôte embarqué à l'intérieur, ce qui irait à l'encontre de l'objectif de se démarquer de l'hôte. Dans cette optique, l'ensemble d'outils cœur est auto-contenue et auto-hébergé. Le reste des paquets du Chapitre 5 se construit contre la nouvelle Glibc de `/tools`.

Avant d'entrer dans l'environnement chroot dans le Chapitre 6, le premier paquet majeur à être installé est Glibc, à cause de sa nature auto-suffisante mentionnée ci-dessus. Une fois que Glibc est installée dans `/usr`, nous allons réaliser une rapide modification des valeurs par défaut de l'ensemble des outils puis continuer la construction du reste du système LFS cible.

5.3. 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 le suivant, mais quelque fois dans seulement un des deux. 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, un point important doit être précisé :

Important

Les instructions de construction supposent que vous avez défini correctement les Host System Requirements, y compris les liens symboliques :

- **bash** is 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/`. *Ne mettez pas* les codes sources dans `/mnt/lfs/tools/`.
2. Allez dans le répertoire des codes sources.
3. Pour chaque paquet :
   a. En utilisant le programme **tar**, décompressez le paquet à construire. Au chapitre 5, assurez-vous d'être l'utilisateur *lfs* lors de l'extraction du paquet.
   b. Allez dans le répertoire créé lorsque le paquet a été décompressé.
   c. Suivez les instructions du livre pour construire le paquet.
   d. Revenez au répertoire des codes sources.
   e. Effacez le répertoire des sources décompressées et tous les répertoires `<paquet>-build` créés pendant le processus de construction, sauf si on vous demande de faire autrement.

5.4. Binutils-2.23.1 - Passe 1

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

Temps de construction estimé : 1 SBU
Espace disque requis : 391 Mo

5.4.1. Installation de Binutils croisé

> **Remarque**
>
> Revenez en arrière et relisez les remarques de la section précédente. La compréhension des remarques notées importantes vous fera éviter beaucoup de problèmes plus tard.

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

La documentation de Binutils recommande de construire Binutils en dehors du répertoire des sources, c'est-à-dire dans un répertoire de construction dédié :

```
mkdir -v ../binutils-build
cd ../binutils-build
```

> **Remarque**
>
> 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 install; }`**.

> **Remarque**
>
> Les valeurs SBU de construction approximatives et de l'espace disque requis au chapitre 5 n'incluent pas les données des suites de tests.

Maintenant, préparez la compilation de Binutils :

```
../binutils-2.23.1/configure     \
    --prefix=/tools            \
    --with-sysroot=$LFS        \
    --with-lib-path=/tools/lib \
    --target=$LFS_TGT          \
    --disable-nls              \
    --disable-werror
```

Voici la signification des options de configure :

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

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

--with-lib-path=/tools/lib
Ceci spécifie le chemin de la bibliothèque qui devrait être configuré dans l'éditeur de liens afin qu'il l'utilise.

--target=$LFS_TGT
Vu que la description de la machine dans la variable LFS_TGT est légèrement différente de la valeur retournée 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 comme 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
```

La compilation est maintenant terminée. Normalement, la suite de tests devrait être lancée mais, à ce moment, l'ensemble de travail de la suite de tests (Tcl, Expect et DejaGnu) n'est pas encore en place. Les bénéfices à lancer les tests maintenant seraient minimes car les programmes de la première passe seront bientôt remplacés par ceux de la seconde.

Si vous construisez sur une x86_64, créez un lien symbolique pour assurer la propreté de notre chaîne d'outils :

```
case $(uname -m) in
  x86_64) mkdir -v /tools/lib && ln -sv lib /tools/lib64 ;;
esac
```

Installez le paquet :

```
make install
```

Les détails sur ce paquet sont disponibles dans Section 6.13.2, « Contenu de Binutils. »

5.5. GCC-4.7.2 - Passe 1

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

Temps de construction estimé :	5.5 SBU
Espace disque requis :	1.4 Go

5.5.1. Installation de GCC croisé

GCC exige maintenant 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 avec GCC. Déballez chaque paquet dans le répertoire du source de GCC et renommez les répertoires ainsi créés pour que les procédures de construction de GCC les utilisent automatiquement :

Remarque

Beaucoup d'incompréhensions existent concernant ce chapitre. Les procédures sont les mêmes que celles des autres chapitres, comme expliqué plus haut (Instructions de construction 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 -Jxf ../mpfr-3.1.1.tar.xz
mv -v mpfr-3.1.1 mpfr
tar -Jxf ../gmp-5.1.0.tar.xz
mv -v gmp-5.1.0 gmp
tar -zxf ../mpc-1.0.1.tar.gz
mv -v mpc-1.0.1 mpc
```

La commande suivante modifiera l'emplacement de l'éditeur de liens dynamique par défaut de GCC pour utiliser celui installé dans `/tools`. Elle supprime aussi `/usr/include` du chemin de recherche include de GCC. Lancez :

```
for file in \
 $(find gcc/config -name linux64.h -o -name linux.h -o -name sysv4.h)
do
  cp -uv $file{,.orig}
  sed -e 's@/lib\(64\)\?\(32\)\?/ld@/tools&@g' \
      -e 's@/usr@/tools@g' $file.orig > $file
  echo '
#undef STANDARD_STARTFILE_PREFIX_1
#undef STANDARD_STARTFILE_PREFIX_2
#define STANDARD_STARTFILE_PREFIX_1 "/tools/lib/"
#define STANDARD_STARTFILE_PREFIX_2 ""' >> $file
  touch $file.orig
done
```

Au cas où ce qui précède vous semble compliqué à suivre, décortiquons-le un peu. On cherche tout d'abord tous les fichiers sous le répertoire `gcc/config` nommés soit `linux.h`, `linux64.h` ou `sysv4.h`. Pour chaque fichier trouvé, on le copie dans un fichier du même nom, mais avec en plus le suffixe « .orig ». Puis la première expression sed envoie « /tools » dans toutes les occurrences « /lib/ld », « /lib64/ld » ou « /lib32/ld », tandis que la deuxième remplace les lignes « /usr » codées en dur. Ensuite, nous ajoutons nos instructions define qui modifient le préfixe des

startfiles par défaut à la fin du fichier. Notez que le suffixe « / » dans « /tools/lib/ » est requis. Enfin, nous utilisons **touch** pour mettre à jour l'horodatage des fichiers copiés. Utilisé avec **cp -u**, ceci empêche des modifications non prévues des fichiers originaux au cas où les commandes sont lancées deux fois par erreur.

GCC ne détecte pas bien la pile de protection, ce qui pose problème pour la construction de Glibc-2.17, corrigez donc cela en lançant la commande suivante :

```
sed -i '/k prot/agcc_cv_libc_provides_ssp=yes' gcc/configure
```

Ne construisez pas les fichiers .info. Ils ne sont pas nécessaires ici et ils sont cassés dans la version actuelle de **makeinfo**.

```
sed -i 's/BUILD_INFO=info/BUILD_INFO=/' gcc/configure
```

La documentation de GCC recommande de ne pas construire GCC dans le répertoire des sources mais dans un répertoire de construction dédié :

```
mkdir -v ../gcc-build
cd ../gcc-build
```

Préparez la compilation de GCC :

```
../gcc-4.7.2/configure         \
    --target=$LFS_TGT          \
    --prefix=/tools            \
    --with-sysroot=$LFS        \
    --with-newlib              \
    --without-headers          \
    --with-local-prefix=/tools \
    --with-native-system-header-dir=/tools/include \
    --disable-nls              \
    --disable-shared           \
    --disable-multilib         \
    --disable-decimal-float    \
    --disable-threads          \
    --disable-libmudflap       \
    --disable-libssp           \
    --disable-libgomp          \
    --disable-libquadmath      \
    --enable-languages=c       \
    --with-mpfr-include=$(pwd)/../gcc-4.7.2/mpfr/src \
    --with-mpfr-lib=$(pwd)/mpfr/src/.libs
```

Voici la signification des options de configure :

`--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 le support 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 emmêche GCC de les chercher.

`--with-local-prefix=/tools`
Le préfixe local est l'emplacement du système où GCC cherchera les fichiers include installés en local. Par défaut, c'est `/usr/local`. Définir ceci sur `/tools` aide à garder l'emplacement de l'hôte `/usr/local` en-dehors du chemin de recherche de GCC.

`--with-native-system-header-dir=/tools/include`
Par défaut, GCC cherche les en-têtes du système dans `/usr/include`. Associé au paramètre sysroot, ceci ne serait pas traduit en `$LFS/usr/include`. Cependant, les en-têtes qui seront installées aux deux prochaines sections iront dans `$LFS/tools/include`. Ce paramètre garantit que gcc les trouvera correctement. Dans la deuxième passe de GCC, ce même paramètre garantira qu'aucune en-tête du système hôte ne soit trouvée.

`--disable-shared`
Ce paramètre oblige GCC à lier ses bibliothèques internes de manière statique. On procède ainsi pour éviter les problèmes avec le système hôte.

`--disable-decimal-float, --disable-threads, --disable-libmudflap, --disable-libssp, --disable-libgomp, --disable-libquadmath`
Ces paramètres désactivent le support de l'extension des virgules flottantes décimales, de threading, libmudflap, libssp et libgomp et libquadmath. Ces fonctionnalités ne parviendront pas à se compiler lors de la construction d'un compilateur croisé et elles ne sont pas nécessaires pour la tâche de compilation croisée de la libc temporaire.

`--disable-multilib`
Sur du x86_64, LFS ne supporte pas encore une configuration multilib (plusieurs bibliothèques). Ce paramètre n'a pas d'importance pour x86.

`--enable-languages=c`
Cette option nous assure que seul le compilateur C sera construit. C'est le seul langage actuellement nécessaire.

Compilez GCC en lançant :

```
make
```

La compilation est maintenant terminée. À ce point, la suite de tests devrait être lancée. Mais, comme nous l'avons dit plus tôt, l'ensemble de travail de la suite de tests n'est pas encore en place. Les bénéfices à lancer les tests maintenant seraient minimes car les programmes de la première passe seront bientôt remplacés.

Installez le paquet :

```
make install
```

L'utilisation de *`--disable-shared`* signifie que le fichier `libgcc_eh.a` n'est pas créé et installé. Le paquet Glibc dépend de cette bibliothèque puisqu'il utilise *`-lgcc_eh`* à l'intérieur de son système de construction. On peut satisfaire cette dépendance en créant un lien symbolique vers `libgcc.a`, puisque ce fichier va finir par contenir les objets normalement contenus dans `libgcc_eh.a`:

```
ln -sv libgcc.a `$LFS_TGT-gcc -print-libgcc-file-name | sed 's/libgcc/&_eh/'`
```

Les détails sur ce paquet sont disponibles dans Section 6.17.2, « Contenu de GCC. »

5.6. Linux-3.8.1 API Headers

Les Linux API Headers (en-têtes API de Linux, incluses dans linux-3.8.1.tar.xz) montrent l'API du noyau pour qu'il soit utilisé par Glibc.

Temps de construction estimé : 0.1 SBU
Espace disque requis : 511 Mo

5.6.1. 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 et d'anciennes dépendances présentes du fait d'une activité précédente :

```
make mrproper
```

Maintenant, testez et faites l'extraction à partir des sources des en-têtes du noyau visibles par l'utilisateur. Elles se situent dans un répertoire local intermédiaire et on les copie dans le répertoire adéquat car le processus d'extraction supprime tous les fichiers existant dans le répertoire cible.

```
make headers_check
make INSTALL_HDR_PATH=dest headers_install
cp -rv dest/include/* /tools/include
```

Les détails sur ce paquet sont situés dans Section 6.7.2, « Contenu de Linux API Headers. »

5.7. Glibc-2.17

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 estimé : 5.4 SBU
Espace disque requis : 554 Mo

5.7.1. Installation de Glibc

Dans certains cas, en particulier avec LFS 7.1, les en-têtes rpc n'étaient pas installées correctement. Regardez si elles sont installées dans le système hôte et, sinon, installez-les :

```
if [ ! -r /usr/include/rpc/types.h ]; then
  su -c 'mkdir -p /usr/include/rpc'
  su -c 'cp -v sunrpc/rpc/*.h /usr/include/rpc'
fi
```

La documentation de Glibc recommande de construire Glibc en dehors du répertoire des sources, c'est-à-dire dans un répertoire dédié :

```
mkdir -v ../glibc-build
cd ../glibc-build
```

Ensuite, préparez la compilation de Glibc :

```
../glibc-2.17/configure                                \
      --prefix=/tools                                  \
      --host=$LFS_TGT                                  \
      --build=$(../glibc-2.17/scripts/config.guess) \
      --disable-profile                                \
      --enable-kernel=2.6.25                           \
      --with-headers=/tools/include                    \
      libc_cv_forced_unwind=yes                        \
      libc_cv_ctors_header=yes                         \
      libc_cv_c_cleanup=yes
```

Voici la signification des options de configure :

`--host=$LFS_TGT, --build=$(../glibc-2.17/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 `/tools`.

`--disable-profile`
Ceci construit les bibliothèques sans les informations de profilage. Enlevez cette option si le profilage sur les outils temporaires est nécessaire.

`--enable-kernel=2.6.25`
Ceci indique à Glibc de compiler la bibliothèque avec le support des noyaux Linux 2.6.25 et supérieurs. Les contournements pour les noyaux plus anciens ne sont pas activés.

`--with-headers=/tools/include`

Ceci dit à Glibc de se compiler contre les en-têtes récemment installées dans le répertoire tools, afin qu'il connaisse exactement les fonctionnalités du noyau et puisse s'optimiser en conséquence.

`libc_cv_forced_unwind=yes`

L'éditeur de liens installé lors de Section 5.4, « Binutils-2.23.1 - Passe 1 » était construit de façon croisée et, dans cet état, il ne peut pas être utilisé tant que Glibc n'a pas été installé. Cela signifie que le test de configure du support force-unwind échouera puisqu'il croit avoir à faire à un éditeur de liens opérationnel. La variable libc_cv_forced_unwind=yes est passée afin d'indiquer à **configure** que le support de force-unwind est disponible sans qu'il n'ait à lancer le test.

`libc_cv_c_cleanup=yes`

De la même façon, nous passons libc_cv_c_cleanup=yes au script **configure** afin que le test soit sauté et que le support de gestion du nettoyage C soit configuré.

`libc_cv_ctors_header=yes`

De la même façon, nous passons libc_cv_ctors_header=yes au script **configure** pour sauter le test et configurer le support du constructeur gcc.

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.

Compilez le paquet :

```
make
```

Ce paquet est fourni avec une suite de test, cependant vous ne pouvez pas l'exécuter à ce moment car nous n'avons pas encore de compilateur C++.

Remarque

La suite de tests exige aussi que des données de locale soient installées afin de s'exécuter avec succès. Les données de locale fournissent au système des informations sur la date, l'heure et les formats normaux acceptés et fournis par les outils systèmes. Si les suites de tests ne sont pas exécutés dans ce chapitre (suivant ainsi notre recommandation), il y a peu d'intérêt à installer les locales maintenant. Les bonnes locales seront installées dans le chapitre suivant. Néanmoins, pour installer les locales Glibc, utilisez les instructions de la section Section 6.9, « Glibc-2.17. »

Installez le paquet :

```
make install
```

Attention

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

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

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: /tools/lib/ld-linux.so.2]
```

Remarquez que `/tools/lib` ou `/tools/lib64` pour les machines 64 bits apparaît comme préfixe de l'éditeur de liens dynamiques.

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

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

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

Remarque

La construction de Binutils dans la section suivante servira de test supplémentaire pour vérifier que l'ensemble d'outils a été construit correctement. Si Binutils échoue pour se construire, c'est une indication que quelque chose ne va pas dans les installations précédentes de Binutils, GCC, ou Glibc.

Les détails sur ce paquet sont situés dans Section 6.9.4, « Contenu de Glibc. »

5.8. Binutils-2.23.1 - Passe 2

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

Temps de construction estimé : 1.1 SBU
Espace disque requis : 407 Mo

5.8.1. Installation de Binutils

Créez de nouveau un répertoire de construction séparé :

```
mkdir -v ../binutils-build
cd ../binutils-build
```

Préparez la compilation de Binutils :

```
CC=$LFS_TGT-gcc           \
AR=$LFS_TGT-ar            \
RANLIB=$LFS_TGT-ranlib    \
../binutils-2.23.1/configure \
    --prefix=/tools       \
    --disable-nls         \
    --with-lib-path=/tools/lib
```

Voici la signification des nouvelles options de configure :

`CC=$LFS_TGT-gcc AR=$LFS_TGT-ar RANLIB=$LFS_TGT-ranlib`
Étant vraiment une construction neuve de Binutils, l'initialisation de ces variables s'assure que le système de construction utilise le compilateur croisé et les outils associés au lieu de ceux du système hôte.

`--with-lib-path=/tools/lib`
Ceci indique au script configure de spécifier le chemin de recherche des bibliothèques lors de la compilation de Binutils, aboutissant au passage de `/tools/lib` à l'éditeur de liens. Ceci empêche l'éditeur de liens de chercher dans tous les répertoires de bibliothèques de l'hôte.

Compilez le paquet :

```
make
```

Installez le paquet :

```
make install
```

Maintenant, préparez l'éditeur de liens pour la phase de « Ré-ajustement » du prochain chapitre :

```
make -C ld clean
make -C ld LIB_PATH=/usr/lib:/lib
cp -v ld/ld-new /tools/bin
```

Voici la signification des paramètres de make :

`-C ld clean`
Ceci dit au programme make de supprimer tous les fichiers construits dans le sous-répertoire `ld`.

`-C ld LIB_PATH=/usr/lib:/lib`

Cette option reconstruit tout dans le sous-répertoire `ld`. La spécification de la variable de Makefile `LIB_PATH` sur la ligne de commande nous permet d'écraser la valeur par défaut du tools temporaire et de pointer vers le bon chemin final. La valeur de cette variable indique le chemin de recherche de la bibliothèque par défaut de l'éditeur de liens. Cette préparation sert pour le chapitre suivant.

Les détails sur ce paquet sont disponibles dans Section 6.13.2, « Contenu de Binutils. »

5.9. GCC-4.7.2 - Passe 2

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

Temps de construction estimé : 7.1 SBU

Espace disque requis : 1.8 Go

5.9.1. Installation de GCC

Notre première construction de GCC a installé deux en-têtes internes au système. En principe, l'une d'entre elles, `limits.h` incluera l'en-tête `limits.h` du système correspondante dans ce cas, `/tools/include/limits.h`. Cependant, au moment de la première construction de gcc, `/tools/include/limits.h` n'existait pas, donc l'en-tête interne qu'a installée GCC est un fichier partiel, à moitié-rempli et il n'inclut pas les fonctions étendues de l'en-tête du système. Ceci s'adaptait à la construcsion de la libc temporaire mais la construction de GCC exige maintenant l'en-tête interne complète. Créez une version complète de l'en-tête interne en utilisant une commande identique à ce que fait le système de construction de GCC en temps normal :

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

Pour les machines x86, une construction bootstrap de GCC utilise le drapeau `-fomit-frame-pointer` du compilateur. Les constructions non-bootstrap suppriment ce drapeau par défaut, l'objectif serait d'obtenir un compilateur qui est exactement le même que si nous étions bootstrapés. Appliquez la commande **sed** suivante pour obliger la construction à utiliser le drapeau :

```
cp -v gcc/Makefile.in{,.tmp}
sed 's/^T_CFLAGS =$/& -fomit-frame-pointer/' gcc/Makefile.in.tmp \
  > gcc/Makefile.in
```

Une fois encore, modifiez l'emplacement de l'éditeur de liens dynamique par défaut de GCC pour utiliser celui installé dans `/tools`.

```
for file in \
 $(find gcc/config -name linux64.h -o -name linux.h -o -name sysv4.h)
do
  cp -uv $file{,.orig}
  sed -e 's@/lib\(64\)\?\(32\)\?/ld@/tools&@g' \
  -e 's@/usr@/tools@g' $file.orig > $file
  echo '
#undef STANDARD_STARTFILE_PREFIX_1
#undef STANDARD_STARTFILE_PREFIX_2
#define STANDARD_STARTFILE_PREFIX_1 "/tools/lib/"
#define STANDARD_STARTFILE_PREFIX_2 ""' >> $file
  touch $file.orig
done
```

Comme dans la première construction de GCC, il a besoin de GMP, de MPFR et MPC. Déballez les archives tar et déplacez-les dans les répertoires nommés comme il le faut :

```
tar -Jxf ../mpfr-3.1.1.tar.xz
mv -v mpfr-3.1.1 mpfr
tar -Jxf ../gmp-5.1.0.tar.xz
mv -v gmp-5.1.0 gmp
tar -zxf ../mpc-1.0.1.tar.gz
mv -v mpc-1.0.1 mpc
```

De nouveau, ne construisez pas les fichiers .info. Ils ne sont pas nécessaires ici et ils sont cassés dans la version actuelle de **makeinfo**.

```
sed -i 's/BUILD_INFO=info/BUILD_INFO=/' gcc/configure
```

Créez un répertoire de construction séparé :

```
mkdir -v ../gcc-build
cd ../gcc-build
```

Avant de commencer la construction de GCC, rappelez-vous de désinitialiser toute variable d'environnement surchargeant les options d'optimisation par défaut.

Maintenant, préparez la compilation de GCC :

```
CC=$LFS_TGT-gcc \
AR=$LFS_TGT-ar                  \
RANLIB=$LFS_TGT-ranlib          \
../gcc-4.7.2/configure          \
    --prefix=/tools             \
    --with-local-prefix=/tools  \
    --with-native-system-header-dir=/tools/include \
    --enable-clocale=gnu        \
    --enable-shared             \
    --enable-threads=posix      \
    --enable-__cxa_atexit       \
    --enable-languages=c,c++    \
    --disable-libstdcxx-pch     \
    --disable-multilib          \
    --disable-bootstrap         \
    --disable-libgomp           \
    --with-mpfr-include=$(pwd)/../gcc-4.7.2/mpfr/src \
    --with-mpfr-lib=$(pwd)/mpfr/src/.libs
```

Voici la signification des nouvelles options de configure :

`--enable-clocale=gnu`

Cette option nous assure que le bon modèle de locale est sélectionné pour les bibliothèques C++ sous toutes les circonstances. Si le script configure trouve la locale *de_DE* installée, il sélectionnera le bon modèle de locale gnu. Néanmoins, si la locale *de_DE* n'est pas installée, il existe un risque de construire des bibliothèques C+

+ incompatibles avec ABI (Application Binary Interface) à cause du choix d'un mauvais modèle générique de locale.

`--enable-threads=posix`

Ceci active la gestion des exceptions C++ pour le code multi-threadé.

`--enable-__cxa_atexit`

Cette option autorise l'utilisation de `__cxa_atexit`, plutôt que `atexit`, pour enregistrer les destructeurs C++ des objets statiques locaux et globaux. Cette option est essentielle pour la gestion des destructeurs en compatibilité complète avec les standards. Il affecte aussi l'ABI C++ et donc résulte en des bibliothèques partagées et des programmes C++ interopérables avec les autres distributions Linux.

`--enable-languages=c,c++`

Cette option garantie que les compilateurs C et C++ seront construits.

`--disable-libstdcxx-pch`

Ce commutateur empêche la construction de l'en-tête précompilé (PCH) de `libstdc++`. Il prend beaucoup d'espace et nous n'en avons aucune utilité.

`--disable-bootstrap`

Pour les constructions natives de GCC, on a par défaut une compilation "bootstrap". Elle ne fait pas que compiler GCC, mais elle le compile plusieurs fois. Elle utilise les programmes compilés dans une première étape pour se compiler une seconde fois, puis une troisième fois à nouveau. Les deuxième et troisième passages sont comparés pour garantir qu'elle peut se reproduire facilement. Cela implique aussi qu'elle a été compilée correctement. Néanmoins, la méthode de compilation LFS devrait fournir un compilateur solide sans qu'il soit nécessaire de bootstraper chaque fois.

Compilez le paquet :

```
make
```

Installez le paquet :

```
make install
```

En touche finale, créez un lien symbolique. Beaucoup de programmes et de scripts lance **cc** au lieu de **gcc**, qui est utilisé pour conserver des programmes génériques, utilisables donc sur n'importe quel type de système où le compilateur C n'est pas toujours installé. L'exécution de **cc** laisse l'administrateur du système décider quel compilateur C installer :

```
ln -sv gcc /tools/bin/cc
```

Attention

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

```
echo 'main(){}' > dummy.c
cc dummy.c
readelf -l a.out | grep ': /tools'
```

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

```
[Requesting program interpreter: /tools/lib/ld-linux.so.2]
```

Remarquez que `/tools/lib` ou `/tools/lib64` pour les machines 64 bits apparaît comme préfixe de l'éditeur de liens dynamiques.

Si l'affichage diffère ou s'il n'y a aucun affichage, alors quelque chose ne se passe pas bien. Enquêtez et tracez vos étapes pour trouver où se cache le problème et comment le corriger. Ce problème doit être corrigé avant de continuer. Tout d'abord, relancez la vérification de propreté en utilisant **gcc** au lieu de **cc**. Si cela fonctionne, le lien symbolique `/tools/bin/cc` est manquant. Installez le lien symbolique comme indiqué ci-dessus. Ensuite, assurez-vous que le `PATH` est correct. Ceci se vérifie en lançant **echo $PATH** et en vérifiant que `/tools/bin` est en tête de la liste. Si le `PATH` est mauvais, cela pourrait signifier que vous n'êtes pas connecté en tant qu'utilisateur `lfs` ou que quelque chose s'est mal passé dans Section 4.4, « Configurer l'environnement. ».

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

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

Les détails sur ce paquet sont situés dans Section 6.17.2, « Contenu de GCC. »

5.10. Tcl-8.6.0

Le paquet Tcl contient le langage de commandes des outils.

Temps de construction estimé : 0.4 SBU
Espace disque requis : 33 Mo

5.10.1. Installation de Tcl

Ce paquet et les trois suivants (Expect, DejaGNU et Check) sont installés uniquement pour supporter les suites de tests de GCC, Binutils et d'autres paquets. Installer quatre paquets dans un but de tests pourrait sembler excessif mais c'est très rassurant, voire essentiel, de savoir que les outils les plus importants fonctionnent correctement. Même si les suites de tests ne sont pas exécutées dans ce chapitre (elles ne sont pas obligatoires), ces paquets sont nécessaires pour lancer les suites de tests du Chapitre 6.

Préparez la compilation de Tcl :

```
cd unix
./configure --prefix=/tools
```

Construisez le paquet :

```
make
```

La compilation est maintenant terminée. Comme décrit plus tôt, l'execution de la suite de tests n'est pas obligatoire pour les outils temporaires dans ce chapitre. Pour lancer cependant la suite de tests de Tcl, faites la commande suivante :

```
TZ=UTC make test
```

Il se peut que la suite de tests de Tcl rencontre des échecs sous certaines conditions concernant l'hôte, conditions qu'on ne comprend pas toujours. Du coup, des échecs de la suite de tests ne sont pas surprenants ici et ne doivent pas être considérés comme critiques. Le paramètre *TZ=UTC* initialise le fuseau horaire avec le temps universel coordonné (*Coordinated Universal Time* soit l'UTC) connu aussi sous le nom de Greenwich Mean Time (GMT), mais seulement pour la durée de l'exécution de la suite de tests. Ceci nous assure que les tests d'horloge fonctionneront correctement. Des détails sur la variable d'environnement TZ sont fournis dans Chapitre 7.

Installez le paquet :

```
make install
```

Autorisez l'écriture dans les bibliothèques installées pour que les symboles de débogage puissent être supprimés plus tard :

```
chmod -v u+w /tools/lib/libtcl8.6.so
```

Installez les en-têtes de Tcl, le prochain paquet, Expect, en a besoin pour se construire.

```
make install-private-headers
```

Maintenant, ajoutez un lien symbolique nécessaire :

```
ln -sv tclsh8.6 /tools/bin/tclsh
```

5.10.2. Contenu de Tcl

Programmes installés: tclsh (lien vers tclsh8.5) et tclsh8.5
Bibliothèque installée: libtcl8.5.so, libtclstub8.5.a

Descriptions courtes

tclsh8.5	Le shell de commandes Tcl
tclsh	Un lien vers tclsh8.5
`libtcl8.5.so`	La bibliothèque Tcl
`libtclstub8.5.a`	La bibliothèque Tcl Stub

5.11. Expect-5.45

Le paquet Expect contient un programme pour réaliser des dialogues scriptés avec d'autres programmes interactifs.

Temps de construction estimé : 0.1 SBU
Espace disque requis : 4.4 Mo

5.11.1. Installation de Expect

Tout d'abord, forcez le script configure de expect à utiliser `/bin/stty` au lieu d'un `/usr/local/bin/stty` qu'il pourrait trouver sur le système hôte. Cela garantira que nos outils de test demeurent propres pour les constructions finales de la chaîne d'outils :

```
cp -v configure{,.orig}
sed 's:/usr/local/bin:/bin:' configure.orig > configure
```

Construisez maintenant le paquet :

```
./configure --prefix=/tools --with-tcl=/tools/lib \
  --with-tclinclude=/tools/include
```

Voici la signification des options de configure :

`--with-tcl=/tools/lib`
Ceci nous assure que le script configure trouve l'installation Tcl dans l'emplacement temporaire des outils à la place d'un résidant sur le système hôte.

`--with-tclinclude=/tools/include`
Ceci indique explicitement à Expect où trouver le répertoire des sources de Tcl et ses en-têtes internes. Utiliser cette option évite certaines conditions d'échec pour **configure** s'il ne peut pas découvrir automatiquement l'emplacement des en-têtes de Tcl.

Construisez le paquet :

```
make
```

La compilation est maintenant terminée. Comme décrit plus tôt, l'execution de la suite de tests n'est pas obligatoire pour les outils temporaires dans ce chapitre. Pour lancer cependant la suite de tests de Expect, faites la commande suivante :

```
make test
```

Remarquez que la suite de tests d'Expect est connue pour avoir de nombreux échecs sous certaines conditions de l'hôte, conditions qui ne sont pas de notre ressort. Du coup, les échecs de la suite de tests ne sont pas surprenants et ne sont pas considérés comme critiques.

Installez-le :

```
make SCRIPTS="" install
```

Voici la signification du paramètre de make :

`SCRIPTS=""`
Ceci empêche l'installation de scripts expect supplémentaires non nécessaires.

5.11.2. Contenu d'Expect

Programme installé: expect
Bibliothèque installée: libexpect-5.45.a

Courte description

expect	Communique avec les autres programmes interactifs suivant un script.
`libexpect-5.45.a`	Contient des fonctions qui permettent à Expect d'être utilisé comme une extension Tcl ou d'être utilisé directement à partir du langage C ou du langage C++ (sans Tcl)

5.12. DejaGNU-1.5

Le paquet DejaGNU contient un ensemble de travail pour tester d'autres programmes.

Temps de construction estimé : moins de 0.1 SBU
Espace disque requis : 4.1 Mo

5.12.1. Installation de DejaGNU

Préparez la compilation de DejaGNU :

```
./configure --prefix=/tools
```

Construisez et installez le paquet :

```
make install
```

Pour tester les résultats, lancez :

```
make check
```

5.12.2. Contenu de DejaGNU

Programme installé: runtest

Courte descriptions

runtest Un script d'emballage qui trouve le bon shell **expect**, puis qui lance DejaGNU

5.13. Check-0.9.9

Check est un environnement de test d'unités de C.

Temps de construction estimé : 0.1 SBU
Espace disque requis : 6.9 Mo

5.13.1. Installation de Check

Préparez la compilation de Check :

```
./configure --prefix=/tools
```

Construisez le paquet :

```
make
```

La compilation est maintenant terminée. Comme indiqué plus haut, le lancement de la suite de tests n'est pas obligatoire pour les outils temporaires de ce chapitre. Pour lancer quand même la suite de tests de Check, exécutez la commande suivante :

```
make check
```

Remarquez que la suite de tests de Check peut mettre pas mal de temps (jusqu'à 4 SBU).

Installez le paquet :

```
make install
```

5.13.2. Contenu de Check

Programme installé: checkmk
Bibliothèque installée: libcheck.{a,so}

Descriptions courtes

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

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

5.14. Ncurses-5.9

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

Temps de construction estimé : 0.5 SBU
Espace disque requis : 35 Mo

5.14.1. Installation de Ncurses

Préparez la compilation de Ncurses :

```
./configure --prefix=/tools --with-shared \
    --without-debug --without-ada --enable-overwrite
```

Voici la signification des options de configure :

`--without-ada`
Ceci nous assure que Ncurses ne construira pas le support du compilateur Ada qui pourrait être présent sur l'hôte mais qui ne sera pas disponible lorsque nous entrerons dans l'environnement **chroot**.

`--enable-overwrite`
Ceci indique à Ncurses d'installer les fichiers d'en-tête dans `/tools/include`, au lieu de `/tools/include/ncurses`, pour s'assurer que les autres paquets trouveront bien les en-têtes de Ncurses.

Compilez le paquet :

```
make
```

Ce paquet a une suite de tests mais elle ne peut être lancée qu'après que le paquet ai été installé. Les tests se trouvent dans le répertoire `test/`. Voir le fichier `README` de ce répertoire pour plus de détails.

Installez le paquet :

```
make install
```

Les détails sur ce paquet sont disponibles dans Section 6.21.2, « Contenu de Ncurses. »

5.15. Bash-4.2

Le paquet Bash contient le shell Bourne-Again.

Temps de construction estimé : 0.4 SBU
Espace disque requis : 48 Mo

5.15.1. Installation de Bash

Tout d'abord, appliquez le correctif suivant pour corriger divers bogues traités en amont :

```
patch -Np1 -i ../bash-4.2-fixes-11.patch
```

Préparez la compilation de Bash :

```
./configure --prefix=/tools --without-bash-malloc
```

Voici la signification des options de configure :

`--without-bash-malloc`
Cette option désactive l'utilisation par 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
```

La compilation est maintenant terminée. Comme décrit plus tôt, l'execution de la suite de tests n'est pas obligatoire pour les outils temporaires dans ce chapitre. Pour lancer cependant la suite de tests de bash, faites la commande suivante :

```
make tests
```

Installez le paquet :

```
make install
```

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

```
ln -sv bash /tools/bin/sh
```

Les détails sur ce paquet sont situés dans Section 6.33.2, « Contenu de Bash. »

5.16. Bzip2-1.0.6

Le paquet Bzip2 contient des programmes de compression et décompression de fichiers. Compresser des fichiers texte avec **bzip2** permet d'atteindre un taux de compression bien meilleur qu'avec l'outil **gzip**.

Temps de construction estimé : moins de 0.1 SBU
Espace disque requis : 5.7 Mo

5.16.1. Installation de Bzip2

Le paquet Bzip2 ne contient pas de script **configure**. Compilez-le et testez-le avec :

```
make
```

Installez le paquet :

```
make PREFIX=/tools install
```

Les détails sur ce paquet sont situés dans Section 6.19.2, « Contenu de Bzip2. »

5.17. Coreutils-8.21

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

Temps de construction estimé : 0.8 SBU
Espace disque requis : 133 Mo

5.17.1. Installation de Coreutils

Préparez la compilation de Coreutils :

```
./configure --prefix=/tools --enable-install-program=hostname
```

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 c'est requis par la suite de tests de Perl.

Compilez le paquet :

```
make
```

La compilation est maintenant terminée. Comme décrit plus tôt, l'exécution de la suite de tests n'est pas obligatoire pour les outils temporaires dans ce chapitre. Pour lancer cependant la suite de tests de Coreutils, faites la commande suivante :

```
make RUN_EXPENSIVE_TESTS=yes check
```

Le paramètre *RUN_EXPENSIVE_TESTS=yes* indique à la suite de tests de lancer quelques tests supplémentaires, considérés relativement coûteux (en terme de puissance CPU et d'utilisation mémoire) mais habituellement sans problème sous Linux.

Installez le paquet :

```
make install
```

Les détails sur ce paquet sont disponibles dans Section 6.27.2, « Contenu de Coreutils. »

5.18. Diffutils-3.2

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

Temps de construction estimé : 0.2 SBU
Espace disque requis : 8.5 Mo

5.18.1. Installation de Diffutils

Corrigez une incompatibilité entre ce paquet et Glibc-2.17 :

```
sed -i -e '/gets is a/d' lib/stdio.in.h
```

Préparez la compilation de Diffutils :

```
./configure --prefix=/tools
```

Compilez le paquet :

```
make
```

La compilation est maintenant terminée. Comme expliqué plus tôt, l'exécution de la suite de tests n'est pas obligatoire pour les outils temporaires dans ce chapitre. Pour lancer cependant la suite de tests de Diffutils, exécutez la commande suivante :

```
make check
```

Installez le paquet :

```
make install
```

Les détails sur ce paquet sont situés dans Section 6.40.2, « Contenu de Diffutils. »

5.19. File-5.13

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

Temps de construction estimé : 0.1 SBU
Espace disque requis : 12.4 Mo

5.19.1. Installation de File

Préparez la compilation de File :

```
./configure --prefix=/tools
```

Compilez le paquet :

```
make
```

La compilation est maintenant terminée. Comme décrit plus tôt, l'exécution de la suite de tests n'est pas obligatoire pour les outils temporaires dans ce chapitre. Pour lancer cependant la suites de tests de File, faites la commande suivante :

```
make check
```

Installez le paquet :

```
make install
```

Les détails sur ce paquet sont disponibles dans Section 6.12.2, « Contenu de File. »

5.20. Findutils-4.4.2

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 estimé : 0.2 SBU
Espace disque requis : 27 Mo

5.20.1. Installation de Findutils

Préparez la compilation de Findutils :

```
./configure --prefix=/tools
```

Compilez le paquet :

```
make
```

La compilation est maintenant terminée. Comme décrit plus tôt, l'exécution de la suite de tests n'est pas obligatoire pour les outils temporaires dans ce chapitre. Pour lancer cependant la suite de tests de Findutils, faites la commande suivante :

```
make check
```

Installez le paquet :

```
make install
```

Les détails sur ce paquet sont situés dans Section 6.42.2, « Contenu de Findutils. »

5.21. Gawk-4.0.2

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

Temps de construction estimé : 0.2 SBU
Espace disque requis : 30 Mo

5.21.1. Installation de Gawk

Préparez la compilation de Gawk :

```
./configure --prefix=/tools
```

Compilez le paquet :

```
make
```

La compilation est maintenant terminée. Comme décrit plus tôt, l'exécution de la suite de tests n'est pas obligatoire pour les outils temporaires dans ce chapitre. Pour lancer cependant la suite de tests de Gawk, faites la commande suivante :

```
make check
```

Installez le paquet :

```
make install
```

Les détails sur ce paquet sont situés dans Section 6.41.2, « Contenu de Gawk. »

5.22. Gettext-0.18.2

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

Temps de construction estimé : 0.6 SBU
Espace disque requis : 101 Mo

5.22.1. Installation de Gettext

Pour notre paramétrage temporaire des outils, nous n'avons besoin de compiler et d'installer qu'un binaire de Gettext.

Préparez la compilation de Gettext :

```
cd gettext-tools
EMACS="no" ./configure --prefix=/tools --disable-shared
```

Voici la signification des options de configure :

`EMACS="no"`
Ceci empêche le script configure de déterminer où installer les fichiers Lisp d'Emacs vu que le test est connu pour échouer sur certains hôtes.

`--disable-shared`
Nous n'avons besoin d'installer aucune bibliothèque partagée de Gettext pour le moment, donc ce n'est pas nécessaire de les compiler.

Compilez le paquet :

```
make -C gnulib-lib
make -C src msgfmt
```

Comme seul un binaire a été compilé, ce n'est pas possible d'exécuter la suite de tests sans compiler des bibliothèques de support complémentaires du paquet Gettext. Il n'est donc pas recommandé d'essayer d'exécuter la suite de tests à cette étape.

Installez le binaire **msgfmt** :

```
cp -v src/msgfmt /tools/bin
```

Les détails sur ce paquet sont situés dans Section 6.44.2, « Contenu de Gettext. »

5.23. Grep-2.14

Le paquet Grep contient des programmes de recherche à l'intérieur de fichiers.

Temps de construction estimé : 0.2 SBU
Espace disque requis : 21 Mo

5.23.1. Installation de Grep

Préparez la compilation de Grep :

```
./configure --prefix=/tools
```

Compilez les programmes :

```
make
```

La compilation est maintenant terminée. Comme décrit plus tôt, l'exécution de la suite de tests n'est pas obligatoire pour les outils temporaires dans ce chapitre. Pour lancer cependant la suite de tests de Grep, faites la commande suivante :

```
make check
```

Installez le paquet :

```
make install
```

Les détails sur ce paquet sont disponibles dans Section 6.31.2, « Contenu de Grep. »

5.24. Gzip-1.5

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

Temps de construction estimé :	0.2 SBU
Espace disque requis :	10 Mo

5.24.1. Installation de Gzip

Préparez la compilation de Gzip :

```
./configure --prefix=/tools
```

Compilez le paquet :

```
make
```

La compilation est maintenant terminée. Comme décrit plus tôt, l'exécution de la suite de tests n'est pas obligatoire pour les outils temporaires dans ce chapitre. Pour lancer cependant la suite de tests de Gzip, faites la commande suivante :

```
make check
```

Installez le paquet :

```
make install
```

Les détails sur ce paquet sont situés dans Section 6.49.2, « Contenu de Gzip. »

5.25. M4-1.4.16

Le paquet M4 contient un processeur de macros.

Temps de construction estimé : 0.2 SBU
Espace disque requis : 16.6 Mo

5.25.1. Installation de M4

Corrigez une incompatibilité entre ce paquet et Glibc-2.17 :

```
sed -i -e '/gets is a/d' lib/stdio.in.h
```

Préparez la compilation de M4 :

```
./configure --prefix=/tools
```

Compilez le paquet :

```
make
```

La compilation est maintenant terminée. Comme décrit plus tôt, l'exécution de la suite de tests n'est pas obligatoire pour les outils temporaires dans ce chapitre. Pour lancer cependant la suite de tests de M4, faites la commande suivante :

```
make check
```

Installez le paquet :

```
make install
```

Les détails de ce paquet sont situés dans Section 6.29.2, « Contenu de M4. »

5.26. Make-3.82

Le paquet Make contient un programme pour compiler des paquets.

Temps de construction estimé :	0.1 SBU
Espace disque requis :	11.2 Mo

5.26.1. Installation de Make

Préparez la compilation de Make :

```
./configure --prefix=/tools
```

Compilez le paquet :

```
make
```

La compilation est maintenant terminée. Comme décrit plus tôt, l'exécution de la suite de tests n'est pas obligatoire pour les outils temporaires dans ce chapitre. Pour lancer cependant la suite de tests de Make, faites la commande suivante :

```
make check
```

Installez le paquet :

```
make install
```

Les détails de ce paquet sont situés dans Section 6.54.2, « Contenu de Make. »

5.27. Patch-2.7.1

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

Temps de construction estimé : 0.1 SBU
Espace disque requis : 3.4 Mo

5.27.1. Installation de Patch

Préparez la compilation de Patch :

```
./configure --prefix=/tools
```

Compilez le paquet :

```
make
```

La compilation est à présent terminée. Comme indiqué plus haut, l'exécution de la suite de tests n'est pas obligatoire dans ce chapitre pour les outils temporaires. Pour lancer néanmoins la suite de tests de Patch, exécutez la commande suivante :

```
make check
```

Installez le paquet :

```
make install
```

Les détails sur ce paquet sont situés dans Section 6.56.2, « Contenu de Patch. »

5.28. Perl-5.16.2

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

Temps de construction estimé : 1.8 SBU
Espace disque requis : 237 Mo

5.28.1. Installation de Perl

Tout d'abord, appliquez la série de correctifs pour adapter certains chemins codés en dur vers la bibliothèque C :

```
patch -Np1 -i ../perl-5.16.2-libc-1.patch
```

Préparez la compilation de Perl :

```
sh Configure -des -Dprefix=/tools
```

Construisez le paquet :

```
make
```

Bien que Perl soit fourni avec une suite de tests, il vaudrait mieux attendre qu'il soit installé au prochain chapitre.

Seuls quelques outils et quelques bibliothèques doivent être installés pour l'instant :

```
cp -v perl cpan/podlators/pod2man /tools/bin
mkdir -pv /tools/lib/perl5/5.16.2
cp -Rv lib/* /tools/lib/perl5/5.16.2
```

Les détails sur ce paquet sont disponibles dans Section 6.37.2, « Contenu de Perl. »

5.29. Sed-4.2.2

Le paquet Sed contient un éditeur de flux.

Temps de construction estimé :	0.1 SBU
Espace disque requis :	10.5 Mo

5.29.1. Installation de Sed

Préparez la compilation de Sed :

```
./configure --prefix=/tools
```

Compilez le paquet :

```
make
```

La compilation est maintenant terminée. Comme décrit plus tôt, l'exécution de la suite de tests n'est pas obligatoire pour les outils temporaires dans ce chapitre. Pour lancer cependant la suite de tests de Sed, faites la commande suivante :

```
make check
```

Installez le paquet :

```
make install
```

Les détails de ce paquet sont situés dans Section 6.18.2, « Contenu de Sed. »

5.30. Tar-1.26

Le paquet Tar contient un programme d'archivage.

Temps de construction estimé :	0.4 SBU
Espace disque requis :	20.6 Mo

5.30.1. Installation de Tar

Corrigez une incompatibilité entre ce paquet et Glibc-2.17 :

```
sed -i -e '/gets is a/d' gnu/stdio.in.h
```

Préparez la compilation de Tar :

```
./configure --prefix=/tools
```

Compilez le paquet :

```
make
```

La compilation est maintenant terminée. Comme décrit plus tôt, l'exécution de la suite de tests n'est pas obligatoire pour les outils temporaires dans ce chapitre. Pour lancer cependant la suite de tests de Tar, faites la commande suivante :

```
make check
```

Installez le paquet :

```
make install
```

Les détails sur ce paquet sont situés dans Section 6.59.2, « Contenu de Tar. »

5.31. Texinfo-5.0

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

Temps de construction estimé : 0.3 SBU
Espace disque requis : 94 Mo

5.31.1. Installation de Texinfo

Préparez la compilation de Texinfo :

```
./configure --prefix=/tools
```

Compilez le paquet :

```
make
```

La compilation est maintenant terminée. Comme décrit plus tôt, l'exécution de la suite de tests n'est pas obligatoire pour les outils temporaires dans ce chapitre. Pour lancer cependant la suite de tests de Texinfo, faites la commande suivante :

```
make check
```

Installez le paquet :

```
make install
```

Les détails de ce paquet sont situés dans Section 6.60.2, « Contenu de Texinfo. »

5.32. Xz-5.0.4

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 estimé : 0.2 SBU
Espace disque requis : 16.3 Mo

5.32.1. Installation de Xz-Utils

Préparez la compilation de Xz :

```
./configure --prefix=/tools
```

Compilez le paquet :

```
make
```

La compilation est maintenant terminée. Comme décrit plus tôt, l'exécution de la suite de tests n'est pas obligatoire pour les outils temporaires dans ce chapitre. Pour lancer cependant la suite de tests de Xz, faites la commande suivante :

```
make check
```

Installez le paquet :

```
make install
```

Les détails sur ce paquet sont situés dans Section 6.46.2, « Contenu de Xz. »

5.33. Supprimer les symboles des fichiers objets

Les étapes de cette section sont optionnelles mais si la partition LFS est plutôt petite, il est intéressant d'apprendre que des éléments inutiles sont supprimables. Les exécutables et les bibliothèques que vous avez construit jusqu'à maintenant contiennent jusqu'à 130 Mo de symboles de débogages inutiles. Supprimez ces symboles avec :

```
strip --strip-debug /tools/lib/*
strip --strip-unneeded /tools/{,s}bin/*
```

Ces commandes vont laisser de côté une vingtaine de fichiers en indiquant qu'elles ne reconnaissent pas leur format. La plupart sont des scripts et non pas des binaires.

Faites attention à ne *pas* utiliser *`--strip-unneeded`* sur les bibliothèques. Cela détruirait les versions statiques et les paquets devraient être de nouveau construits.

Pour sauver encore davantage, supprimez toute la documentation :

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

Il y aura maintenant au moins 850 Mo d'espace disque libre sur le système de fichiers `$LFS` à utiliser pour construire et installer Glibc dans la prochaine phase. Si vous pouvez construire et installer Glibc, vous pourrez aussi construire et installer le reste.

5.34. Changer de propriétaire

Remarque

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

Pour l'instant, le répertoire `$LFS/tools` appartient à l'utilisateur `lfs`, un utilisateur qui n'existe que sur le système hôte. Si le répertoire `$LFS/tools` reste ainsi, les fichiers appartiennent à un ID utilisateur sans compte correspondant. C'est dangereux car un compte utilisateur créé plus tard pourrait se voir attribuer ce même ID utilisateur et être propriétaire du répertoire `$LFS/tools` et de tous les fichiers à l'intérieur, les exposant ainsi à des manipulations suspectes.

Pour éviter ce problème, vous pourriez ajouter l'utilisateur `lfs` au nouveau système LFS plus tard lorsque vous créeriez le fichier `/etc/passwd`, en prenant garde à assigner les ID utilisateur et groupe de la même manière que sur le Système hôte. Mieux encore, changez le propriétaire du répertoire `$LFS/tools` en le rendant à l'utilisateur `root` en exécutant les commandes suivantes :

```
chown -R root:root $LFS/tools
```

Bien que le dossier `$LFS/tools` puisse être effacé quand la construction du système sera fini, il peut être conservé pour construire des systèmes LFS supplémentaires *de la même version du livre*. La meilleure façon de sauvegarder `$LFS/tools` est celle qui correspond à vos préférences personnelles.

Attention

Si vous souhaitez conserver les outils temporaires pour un usage dans la construction de futurs systèmes LFS, c'est le moment *à présent* de les sauvegarder. Les commandes qu'implique le chapitre 6 vont modifier les outils actuellement en place, les rendant inutiles pour de futures constructions.

Partie III. Construction du système LFS

Chapitre 6. Installer les logiciels du système de base

6.1. Introduction

Dans ce chapitre, nous entrons dans le site de construction et lançons la construction du système LFS. Autrement dit, nous entrons avec chroot dans le mini système Linux temporaire, faisons quelques préparations finales et lançons l'installation de tous les paquets un par un.

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.

L'ordre dans lequel les paquets sont installés dans ce chapitre a besoin d'être strictement suivi pour s'assurer qu'aucun programme n'acquiert accidentellement un chemin ayant comme référence `/tools` en dur. Pour la même raison, ne compilez pas des paquets en parallèle. La compilation en parallèle peut permettre de gagner du temps (tout particulièrement sur les machines à deux CPU), mais cela pourrait résulter en un programme contenant un chemin codé en dur vers `/tools`, ce qui fera arrêter le programme de fonctionner si ce répertoire est supprimé.

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 les autres paquets nécessaires lors de cette étape de construction. Suivant les instructions d'installation, il existe une liste de programmes et de bibliothèques (avec quelques brèves descriptions de ceux-ci) que le paquet installe.

Remarque

Les valeurs SBU et l'espace disque requis incluent les données de suites de tests pour tous les paquets du chapitre 6 auxquels elles sont applicables.

6.2. 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 -v $LFS/{dev,proc,sys}
```

6.2.1. Création des noeuds initiaux vers les périphériques

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

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

6.2.2. Monter et peupler /dev

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

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

6.2.3. Monter les systèmes de fichiers virtuels du noyau

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

```
mount -vt devpts devpts $LFS/dev/pts
mount -vt proc proc $LFS/proc
mount -vt sysfs sysfs $LFS/sys
```

Dans certains systèmes hôtes, `/dev/shm` est un lien symbolique vers `/run/shm`. Dans un environnement chroot, il faut monter ce fichier temporaire hors du système de fichiers de l'hôte :

```
if [ -h $LFS/dev/shm ]; then
   link=$(readlink $LFS/dev/shm)
   mkdir -p $LFS/$link
   mount -vt tmpfs shm $LFS/$link
   unset link
else
   mount -vt tmpfs shm $LFS/dev/shm
fi
```

6.3. 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.

6.3.1. 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 les nouvelles versions. Voici quelques points à connaître pour une mise à jour de paquets, spécifiquement sur un système en cours de fonctionnement

- Il est recommandé, si un des outils de l'ensemble des outils (glibc, gcc, binutils) doit être mis à jour avec une nouvelle version mineure, 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. Par exemple, si glibc-2.2.x a besoin d'être mis à jour vers glibc-2.3.x, il est préférable de reconstruire. Pour les mises à jour encore plus mineures, une simple réinstallation fonctionne généralement mais cela n'est pas garanti. Par exemple, mettre à jour de glibc-2.3.1 à glibc-2.3.2 ne causera aucun problème.
- Si un paquet contenant une bibliothèque partagée est mise 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`. Remarquez que vous ne devez pas supprimer les anciennes bibliothèques jusqu'à ce que les paquets indépendants soient recompilés.

6.3.2. 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.

6.3.2.1. 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.

6.3.2.2. 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 que quelques paquets, ce schéma devient ingérable.

6.3.2.3. 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 supportent 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.

6.3.2.4. 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 ne soit 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.

6.3.2.5. 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-même 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'utiliier **strace**, qui trace tous les appels du système faits pendant l'exécution des scripts d'installation.

6.3.2.6. 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 sur 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://lfs.traduc.org/view/astuces/fakeroot.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*.

6.3.2.7. Gestion basée sur les utilisateurs

Ce schéma, unique à LFS, a été décrit par Matthias Benkmann et est disponible sur le *Projet des astuces*. Dans ce schéma, 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 fonctionnalités et avantages de cette approche sont trop complexes pour les décrire dans cette section. Pour plus de détails, voir l'astuce sur *http://lfs.traduc.org/view/astuces/gestionnaire-paquets-utilisateur.txt*.

6.3.3. 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 transfère 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`, and `/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.

Enfin, vous devez rendre le nouveau système amorçable via Section 8.4, « Utiliser GRUB pour paramétrer le processus de démarrage ».

6.4. Entrer dans l'environnement chroot

Il est temps d'entrer dans l'environnement chroot pour commencer la construction et l'installation du système final LFS. En tant que `root`, lancez la commande suivante pour entrer dans ce petit monde peuplé seulement, pour le moment, des outils temporaires :

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

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

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

Remarquez que `/tools/bin` arrive dernier dans le `PATH`. Ceci signifie qu'un outil temporaire ne sera plus utilisé une fois que la version finale sera installée. Ceci survient quand le shell ne se « rappelle » plus des emplacements des binaires exécutés— Pour cette raison, le hachage est désactivé en passant l'option *`+h`* à **bash**.

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

Remarque

Il est important que toutes les commandes pour le reste de ce chapitre et les chapitres suivants soient lancées à l'intérieur de l'environnement chroot. Si vous devez quitter cet environnement pour une quelconque raison (un redémarrage par exemple), vous devez vous rappeler de commencer par monter les systèmes de fichiers comme expliqué aux Section 6.2.2, « Monter et peupler /dev » et Section 6.2.3, « Monter les systèmes de fichiers virtuels du noyau » entrer de nouveau dans chroot avant de continuer les installations.

6.5. Créer les répertoires

Il est temps de créer la hiérarchie de répertoires sur le système de fichiers LFS. Créez une hiérarchie de répertoires standards en lançant les commandes suivantes :

```
mkdir -pv /{bin,boot,etc/{opt,sysconfig},home,lib,mnt,opt,run}
mkdir -pv /{media/{floppy,cdrom},sbin,srv,var}
install -dv -m 0750 /root
install -dv -m 1777 /tmp /var/tmp
mkdir -pv /usr/{,local/}{bin,include,lib,sbin,src}
mkdir -pv /usr/{,local/}share/{doc,info,locale,man}
mkdir -v  /usr/{,local/}share/{misc,terminfo,zoneinfo}
mkdir -pv /usr/{,local/}share/man/man{1..8}
for dir in /usr /usr/local; do
  ln -sv share/{man,doc,info} $dir
done
case $(uname -m) in
 x86_64) ln -sv lib /lib64 && ln -sv lib /usr/lib64 ;;
esac
mkdir -v /var/{log,mail,spool}
ln -sv /run /var/run
ln -sv /run/lock /var/lock
mkdir -pv /var/{opt,cache,lib/{misc,locate},local}
```

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

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

6.5.1. Remarques à propos de la conformité FHS

L'arborescence de répertoires est basée sur le standard FHS (Filesystem Hierarchy Standard), disponible sur *http://www.pathname.com/fhs/*. Outre le FHS, nous créons des liens symboliques pour la compatibilité pour les répertoires `man`, `doc`, et `info` vu que beaucoup de paquets essaient encore d'installer leur documentation dans `/usr/<répertoire>` ou `/usr/local/<répertoire>` au lieu de `/usr/share/<répertoire>` ou `/usr/local/share/<répertoire>`. Le FHS stipule aussi l'existence de `/usr/local/games` et `/usr/share/games`. Le FHS n'est pas précis en ce qui concerne la structure du sous-répertoire `/usr/local/share`, donc nous créons seulement les répertoires nécessaires. Néanmoins, n'hésitez pas à créer ces répertoires si vous préférez vous conformer plus strictement au FHS.

6.6. Créer les fichiers et les liens symboliques essentiels

Certains programmes stockent en dur des chemins vers des programmes qui n'existent pas encore. Pour satisfaire ces programmes, créez un certain nombre de liens symboliques qui seront remplacés par les vrais fichiers tout au long de ce chapitre une fois que tous les logiciels seront installés :

```
ln -sv /tools/bin/{bash,cat,echo,pwd,stty} /bin
ln -sv /tools/bin/perl /usr/bin
ln -sv /tools/lib/libgcc_s.so{,.1} /usr/lib
ln -sv /tools/lib/libstdc++.so{,.6} /usr/lib
sed 's/tools/usr/' /tools/lib/libstdc++.la > /usr/lib/libstdc++.la
ln -sv bash /bin/sh
```

Un bon système Linux garde une liste des systèmes de fichiers montés dans le fichier `/etc/mtab`. Normalement, ce fichier est créé quand un nouveau système de fichiers est monté. Comme nous ne monterons aucun système de fichiers dans notre environnement chroot, créez un fichier vide pour les utilitaires qui s'attendent à la présence de `/etc/mtab` :

```
touch /etc/mtab
```

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

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

```
cat > /etc/passwd << "EOF"
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/dev/null:/bin/false
nobody:x:99:99:Unprivileged User:/dev/null:/bin/false
EOF
```

Le mot de passe actuel pour `root` (le « x » utilisé est seulement un exemple) sera paramétré plus tard.

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

```
cat > /etc/group << "EOF"
root:x:0:
bin:x:1:
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:
mail:x:34:
nogroup:x:99:
EOF
```

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

Pour supprimer l'invite « I have no name! », démarrez un nouveau shell. Comme nous avons installé une Glibc complète dans le Chapitre 5 et créé les fichiers `/etc/passwd` et `/etc/group`, la résolution du nom d'utilisateur et de groupe fonctionnera à présent :

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

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

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

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

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

Remarque

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

6.7. Linux-3.8.1 API Headers

Les Linux API Headers (en-têtes API de Linux, incluses dans linux-3.8.1.tar.xz) montrent l'API du noyau pour qu'il soit utilisé par Glibc.

Temps de construction estimé : 0.1 SBU
Espace disque requis : 515 Mo

6.7.1. 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 se fait en nettoyant les fichiers d'en-tête C qui sont contenus dans l'archive de la source du noyau Linux.

Assurez-vous qu'il n'y a pas de vieux fichiers et d'anciennes dépendances présentes du fait d'une activité précédente :

```
make mrproper
```

Maintenant, testez et faites l'extraction à partir des sources des en-têtes du noyau visibles par l'utilisateur. Elles se situent dans un répertoire local intermédiaire et on les copie dans le répertoire adéquat car le processus d'extraction supprime tous les fichiers existant dans le répertoire cible. Certains fichiers cachés utilisés par les développeurs du noyau et inutiles dans LFS, sont supprimés du répertoire intermédiaire.

```
make headers_check
make INSTALL_HDR_PATH=dest headers_install
find dest/include \( -name .install -o -name ..install.cmd \) -delete
cp -rv dest/include/* /usr/include
```

6.7.2. Contenu de Linux API Headers

En-têtes installées: /usr/include/asm/*.h, /usr/include/asm-generic/*.h, /usr/include/drm/*.h, /usr/include/linux/*.h, /usr/include/mtd/*.h, /usr/include/rdma/*.h, /usr/include/scsi/*.h, /usr/include/sound/*.h, /usr/include/video/*.h, /usr/include/xen/*.h
Répertoires installés: /usr/include/asm, /usr/include/asm-generic, /usr/include/drm, /usr/include/linux, /usr/include/mtd, /usr/include/rdma, /usr/include/scsi, /usr/include/sound, /usr/include/video, /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 génériques ASM 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 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 Linux
`/usr/include/sound/*.h`	Les en-têtes sons de l'API de Linux

`/usr/include/video/*.h`	Les en-têtes vidéos de l'API de Linux
`/usr/include/xen/*.h`	Les en-têtes Xen de l'API Linux

6.8. Man-pages-3.46

Le paquet Man-pages contient environ 1 900 pages de manuel.

Temps de construction estimé : moins de 0.1 SBU
Espace disque requis : 22 Mo

6.8.1. Installation de Man-pages

Installez Man-pages en lançant :

```
make install
```

6.8.2. Contenu de Man-pages

Fichiers installés: différentes pages de manuel

Descriptions courtes

`pages man` Décrivent les fonctions C et C++, les fichiers périphériques importants et des fichiers de configuration significatifs

6.9. Glibc-2.17

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 estimé : 17.6 SBU
Espace disque requis : 852 Mo

6.9.1. Installation de Glibc

Remarque

Certains paquets non compris dans LFS suggèrent d'installer GNU libiconv pour traduire les données d'un codage en un autre. La page d'accueil du projet (*http://www.gnu.org/software/libiconv/*) précise « Cette bibliothèque fournit une implémentation de `iconv()` à utiliser sur les systèmes qui n'en disposent pas ou dont l'implémentation ne convertit pas l'Unicode. » Glibc fournit une implémentation d'`iconv()` et peut convertir de l'Unicode, du coup libiconv n'est pas requis sur un système LFS.

Le système de construction de Glibc est auto-extractible et il s'installera parfaitement, même si le fichier specs du compilateur et l'éditeur de liens pointent vers `/tools`. Les specs et l'éditeur de liens ne peuvent pas être ajustés avant l'installation de Glibc, car les tests autoconf de Glibc donneraient de faux résultats, ce qui irait à l'encontre du but de faire une construction propre.

Corrigez un bogue qui fait échouer la construction dans l'environnement LFS :

```
sed -i 's#<rpc/types.h>#"rpc/types.h"#' sunrpc/rpc_clntout.c
```

La documentation de Glibc recommande de construire Glibc en dehors du répertoire des sources dans un répertoire de construction dédié :

```
mkdir -v ../glibc-build
cd ../glibc-build
```

Préparez la compilation de Glibc :

```
../glibc-2.17/configure  \
    --prefix=/usr          \
    --disable-profile      \
    --enable-kernel=2.6.25 \
    --libexecdir=/usr/lib/glibc
```

Voici la signification des options de configure :

`--libexecdir=/usr/lib/glibc`
Ceci modifie l'emplacement du programme **pt_chown**, par défaut `/usr/libexec`, par `/usr/lib/glibc`.

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. Maintenant, testez les résultats de la construction :

```
make -k check 2>&1 | tee glibc-check-log
grep Error glibc-check-log
```

Vous verrez probablement un échec attendu (ignoré) lors des tests de *posix/annexc* et de *conform/run-conformtest*. En outre, la suite de tests de Glibc est quelque peu dépendante du système hôte. Voici une liste des problèmes les plus fréquents :

- Les tests *nptl/tst-attr3*, *tst/tst-cputimer1*, et *rt/tst-cpuclock2* sont connus pour échouer. On n'a pas encore totalement compris la raison, mais des indications laissent penser que des problèmes mineurs de temps peuvent être à l'origine de ces échecs.
- Le test math échoue parfois lorsqu'il s'exécute sur des systèmes ou le processeur n'est pas complètement de type Intel ou avec un processeur authentique AMD.
- Lors d'une exécution sur un matériel ancien et lent, quelques tests peuvent échouer à cause de délais de test dépassés. La modification de la commande make check pour paramétrer un TIMEOUTFACTOR a été signalée comme aidant à éliminer ces erreurs (par exemple **TIMEOUTFACTOR=16 make -k check**).
- D'autres tests qui sont connus pour échouer sur certaines architectures sont posix/bug-regex32, misc/tst-writev, elf/check-textrel, nptl/tst-getpid2, et stdio-common/bug22.

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
```

Installez le paquet :

```
make install
```

Installez les en-têtes liées à NIS et à RPC qui ne sont pas installées par défaut ; elles sont requises pour reconstruire glibc et pour plusieurs paquets de BLFS :

```
cp -v ../glibc-2.17/sunrpc/rpc/*.h /usr/include/rpc
cp -v ../glibc-2.17/sunrpc/rpcsvc/*.h /usr/include/rpcsvc
cp -v ../glibc-2.17/nis/rpcsvc/*.h /usr/include/rpcsvc
```

Les locales qui permettent à votre système de répondre en une langue différente n'ont pas été installées avec la commande ci-dessus. Aucune n'est indispensable, mais si certaines sont absentes, les suites de test des futurs paquets peuvent sauter des situations de test importantes.

Vous pouvez installer les locales individuelles en utilisant le programme **localedef** Par exemple, la première commande **localedef** ci-dessous combine la définition de la locale du codage indépendant `/usr/share/i18n/locales/cs_CZ` avec la définition de la page de codes `/usr/share/i18n/charmaps/UTF-8.gz` et envoie le résultat vers le fichier `/usr/lib/locale/locale-archive`. Les instructions suivantes installeront les paramètres minimums des locales nécessaires pour le déroulement optimal des tests :

```
mkdir -pv /usr/lib/locale
localedef -i 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 en_GB -f UTF-8 en_GB.UTF-8
localedef -i en_HK -f ISO-8859-1 en_HK
localedef -i en_PH -f ISO-8859-1 en_PH
localedef -i en_US -f ISO-8859-1 en_US
localedef -i en_US -f UTF-8 en_US.UTF-8
localedef -i es_MX -f ISO-8859-1 es_MX
localedef -i fa_IR -f UTF-8 fa_IR
localedef -i fr_FR -f ISO-8859-1 fr_FR
localedef -i fr_FR@euro -f ISO-8859-15 fr_FR@euro
localedef -i fr_FR -f UTF-8 fr_FR.UTF-8
localedef -i it_IT -f ISO-8859-1 it_IT
localedef -i it_IT -f UTF-8 it_IT.UTF-8
localedef -i ja_JP -f EUC-JP ja_JP
localedef -i ru_RU -f KOI8-R ru_RU.KOI8-R
localedef -i ru_RU -f UTF-8 ru_RU.UTF-8
localedef -i tr_TR -f UTF-8 tr_TR.UTF-8
localedef -i zh_CN -f GB18030 zh_CN.GB18030
```

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-2.17/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-2.17/localedata/SUPPORTED` dans le cas peu probable où vous en auriez besoin.

6.9.2. Configurer Glibc

Le fichier `/etc/nsswitch.conf` doit être créé parce que, bien que Glibc en fournisse un par défaut lorsque ce fichier est manquant ou corrompu, les valeurs par défaut de Glibc ne fonctionnent pas bien dans un environnement en réseau. De plus, le fuseau horaire a besoin d'être configuré.

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
```

Installez timezone data (les données de fuseau horaire) :

```
tar -xf ../tzdata2012j.tar.gz

ZONEINFO=/usr/share/zoneinfo
mkdir -pv $ZONEINFO/{posix,right}

for tz in etcetera southamerica northamerica europe africa antarctica  \
          asia australasia backward pacificnew solar87 solar88 solar89 \
          systemv; do
    zic -L /dev/null   -d $ZONEINFO       -y "sh yearistype.sh" ${tz}
    zic -L /dev/null   -d $ZONEINFO/posix -y "sh yearistype.sh" ${tz}
    zic -L leapseconds -d $ZONEINFO/right -y "sh yearistype.sh" ${tz}
done

cp -v zone.tab iso3166.tab $ZONEINFO
zic -d $ZONEINFO -p Europe/Paris
unset ZONEINFO
```

Voici la signification de la commande zic :

`zic -L /dev/null ...`

Ceci crée des fuseaux horaires posix, sans sauts de secondes. 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 1.9Mio en n'utilisant pas le répertoire `posix`, mais certaines applications ou suites de tests pourraient donner de bons résultats

zic -L leapseconds ...

Ceci crée de bons fuseaux horaires incluant les décalages de secondes. 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 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 :

```
cp -v --remove-destination /usr/share/zoneinfo/<xxx> \
    /etc/localtime
```

Remplacez *<xxx>* par le nom du fuseau horaire sélectionné (par exemple America/Montreal).

Voici la signification de l'option de cp :

--remove-destination

Ceci est nécessaire pour forcer la suppression du lien symbolique déjà existant. La raison pour laquelle nous copions plutôt que de simplement créer un lien symbolique est de se couvrir de la situation où `/usr` serait une partition séparée. Ceci pourrait arriver, par exemple, en démarrant en mode utilisateur unique.

6.9.3. 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 /etc/ld.so.conf.d
```

6.9.4. Contenu de Glibc

Programmes installés: catchsegv, gencat, getconf, getent, iconv, iconvconfig, ldconfig, ldd, lddlibc4, locale, localedef, makedb, mtrace, nscd, pcprofiledump, pldd, pt_chown, rpcgen, sln, sotruss, sprof, tzselect, xtrace, zdump et zic

Bibliothèques installées: ld.so, libBrokenLocale.{a,so}, libSegFault.so, libanl.{a,so}, libbsd-compat.a, libc.{a,so}, libc_nonshared.a, libcidn.so, libcrypt.{a,so}, libdl.{a,so}, libg.a, libieee.a, libm.{a,so}, libmcheck.a, libmemusage.so, libnsl.{a,so}, libnss_compat.so, libnss_dns.so, libnss_files.so, libnss_hesiod.so, libnss_nis.so, libnss_nisplus.so, libpcprofile.so, libpthread.{a,so}, libpthread_nonshared.a, libresolv.{a,so}, librpcsvc.a, librt.{a,so}, libthread_db.so et libutil.{a,so}

Répertoires installés: /usr/include/arpa, /usr/include/bits, /usr/include/gnu, /usr/include/net, /usr/include/netash, /usr/include/netatalk, /usr/include/netax25, /usr/include/neteconet, /usr/include/netinet, /usr/include/netipx, /usr/include/netiucv, /usr/include/netpacket, /usr/include/netrom, /usr/include/netrose, /usr/include/nfs, /usr/include/protocols, /usr/include/rpc, /usr/include/rpcsvc, /usr/include/sys, /usr/lib/audit, /usr/lib/gconv, /usr/lib/glibc, /usr/lib/locale, /usr/share/i18n, /usr/share/zoneinfo, /var/db

Descriptions courtes

catchsegv Peut être utilisé pour créer une trace de la pile lorsqu'un programme s'arrête avec une erreur de segmentation

gencat Génère des catalogues de messages

getconf Affiche les valeurs de configuration du système pour les variables spécifiques du système de fichiers

getent Récupère les entrées à partir d'une base de données administrative

iconv Réalise une conversion de l'ensemble des caractères

iconvconfig Crée des fichiers de configuration pour le module **iconv**

ldconfig Configure les liens du chargeur dynamique

ldd Indique les bibliothèques partagées requises pour chaque programme ou bibliothèque partagée

lddlibc4 Assiste **ldd** avec des fichiers objets

locale Affiche diverses informations sur la locale courante

localedef Compile les spécifications de locale

makedb	Crée une base de données simple à partir d'une entrée textuelle
mtrace	Lit et interprète un fichier de trace mémoire et affiche un résumé dans un format lisible par un humain
nscd	Un démon pour les services de noms fournissant un cache pour les requêtes les plus communes
pcprofiledump	Affiche des informations générées par un profilage du PC
pldd	Liste les objets dynamiques partagés utilisés en exécutant des processus
pt_chown	Un programme d'aide pour que **grantpt** initialise les droits des propriétaires, groupes et autres d'un pseudo-terminal esclave
rpcgen	Génère du code C pour implémenter le protocole RPC (*Remote Procedure Call*)
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-2.17/locale/broken_cur_max.c` pour plus d'informations
`libSegFault`	Un gestionnaire de signaux d'erreurs de segmentation, utilisé par **catchsegv**
`libanl`	Une bibliothèque asynchrone de recherche de noms
`libbsd-compat`	Fournit la portabilité nécessaire pour faire fonctionner certains programmes BSD (Berkeley Software Distribution) sous Linux
`libc`	La principale bibliothèque C
`libcidn`	Utilisé en interne par Glibc pour la gestion des noms de domaine internationalisés dans la fonction `getaddrinfo()`
`libcrypt`	La bibliothèque de chiffrement
`libdl`	La bibliothèque de l'interface du chargeur dynamique
`libg`	Bibliothèque factice ne contenant aucune fonction. C'était auparavant une bibliothèque d'exécution pour **g++**
`libieee`	Un lien vers ce module provoque volontairement des règles de gestion d'erreur pour les fonctions math telles que définies par les *Institute of Electrical and Electronic Engineers* (IEEE). Le paramètre par défaut est la gestion de l'erreur POSIX.1
`libm`	La bibliothèque mathématique
`libmcheck`	Active le test d'allocation de mémoire lorsqu'on y relie quelque chose

`libmemusage`	Utilisé par **memusage** pour aider à la récupération d'informations sur l'utilisation de la mémoire par un programme
`libnsl`	La bibliothèque de services réseau
`libnss`	Les bibliothèques « Name Service Switch », contenant des fonctions de résolution de noms d'hôtes, de noms d'utilisateurs, de noms de groupes, d'alias, de services, de protocoles et ainsi de suite
`libpcprofile`	Contient des fonctions de profilage utilisées pour tracer le temps CPU dépensé sur les lignes de code source
`libpthread`	La bibliothèque threads POSIX
`libresolv`	Contient des fonctions de création, d'envoi et d'interprétation de paquets pour les serveurs de noms de domaine Internet
`librpcsvc`	Contient des fonctions apportant différents services RPC
`librt`	Contient des fonctions fournissant la plupart des interfaces spécifiées par l'extension temps réel de POSIX.1b
`libthread_db`	Contient des fonctions utiles pour construire des débogueurs de programmes multi-threads
`libutil`	Contient du code pour les fonctions « standard » utilisées par de nombreux outils Unix

6.10. Ajustement de la chaîne d'outils

Maintenant que les bibliothèques C finales ont été installées, il est temps d'ajuster la chaîne d'outils pour qu'elle lie tout programme nouvellement compilé à ces nouvelles bibliothèques.

D'abord, sauvegardez l'éditeur de liens de `/tools`, et remplacez-le par l'éditeur de lien ajusté que nous avons fait au chapitre 5. Nous créerons aussi un lien vers son équivalent dans `/tools/$(gcc -dumpmachine)/bin` :

```
mv -v /tools/bin/{ld,ld-old}
mv -v /tools/$(gcc -dumpmachine)/bin/{ld,ld-old}
mv -v /tools/bin/{ld-new,ld}
ln -sv /tools/bin/ld /tools/$(gcc -dumpmachine)/bin/ld
```

Puis, modifiez le fichier des specs GCC afin qu'il pointe vers le nouvel éditeur de liens dynamiques. La suppression simple de tous les exemples de « /tools » devrait nous laisser uniquement le bon chemin sur l'éditeur de liens dynamique. Ajustez aussi le fichier de specs pour que GCC sache où trouver les en-têtes corrects et les fichiers de démarrage de Glibc. Une commande **sed** fait cela :

```
gcc -dumpspecs | sed -e 's@/tools@@g' \
    -e '/\*startfile_prefix_spec:/{n;s@.*@/usr/lib/ @}' \
    -e '/\*cpp:/{n;s@$@ -isystem /usr/include@}' > \
    `dirname $(gcc --print-libgcc-file-name)`/specs
```

C'est une bonne idée d'examiner visuellement le fichier de specs pour vérifier que le changement voulu a bien été effectué.

Il est impératif à ce moment d'arrêter et de vous assurer que les fonctions basiques (compilation et édition des liens) de l'ensemble des outils ajusté fonctionnent comme attendu. Pour cela, réalisez une petite vérification :

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

Si tout fonctionne correctement, il ne devrait pas y avoir d'erreur et la sortie de la commande sera (avec des différences spécifiques aux plateformes dans le nom de l'éditeur de liens) :

```
[Requesting program interpreter: /lib/ld-linux.so.2]
```

Remarquez que `/lib` est maintenant le préfixe de notre éditeur de liens.

Maintenant, assurez-vous que nous utilisons les bons fichiers de démarrage :

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

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

```
/usr/lib/crt1.o succeeded
/usr/lib/crti.o succeeded
/usr/lib/crtn.o succeeded
```

Vérifiez que le compilateur cherche les bons fichiers d'en-tête :

```
grep -B1 '^ /usr/include' dummy.log
```

Cette commande devrait réussir avec la sortie suivante :

```
#include <...> search starts here:
 /usr/include
```

Puis, vérifiez que le nouvel éditeur de liens est utilisé avec les bons chemins de recherche :

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

Si tout fonctionne correctement, il ne devrait pas y avoir d'erreur et la sortie de la dernière commande sera (selon la triplette cible spécifique à chaque plateforme) :

```
SEARCH_DIR("/tools/i686-pc-linux-gnu/lib")
SEARCH_DIR("/usr/lib")
SEARCH_DIR("/lib");
```

Ensuite, assurez-vous que nous utilisons la bonne libc :

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

Si tout fonctionne correctement, il ne devrait pas y avoir d'erreur et la sortie de la dernière commande sera (selon la triplette cible spécifique à chaque plateforme) :

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

Pour finir, assurez-vous que GCC utilise le bon éditeur de liens dynamiques :

```
grep found dummy.log
```

Si tout fonctionne correctement, il ne devrait pas y avoir d'erreur et la sortie de la commande sera (avec des différences spécifiques aux plateformes dans le nom de l'éditeur de liens et un répertoire lib64 sur les hôtes 64 bits) :

```
found ld-linux.so.2 at /lib/ld-linux.so.2
```

Si la sortie n'apparaît pas comme montré ci-dessus ou qu'elle n'apparaît pas du tout, alors quelque chose ne va vraiment pas. Enquêtez et retracez les étapes pour savoir d'où vient le problème et comment le corriger. La raison la plus probable est que quelque chose s'est mal passé lors de la modification du fichier specs ci-dessus. Tout problème devra être résolu avant de continuer le processus.

Une fois que tout fonctionne correctement, nettoyez les fichiers tests :

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

6.11. Zlib-1.2.7

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

Temps de construction estimé : 0.1 SBU
Espace disque requis : 3.9 Mo

6.11.1. Installation de Zlib

Préparez la compilation de Zlib :

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

Compilez le paquet :

```
make
```

Pour tester les résultats, lancez :

```
make check
```

Installez le paquet :

```
make install
```

La bibliothèque partagée doit être déplacée vers `/lib`, et il s'en suit donc que `/usr/lib` devra être recréé :

```
mv -v /usr/lib/libz.so.* /lib
ln -sfv ../../lib/libz.so.1.2.7 /usr/lib/libz.so
```

6.11.2. Contenu de Zlib

Bibliothèques installées: libz.{a,so}

Descriptions courtes

`libz` Contient des fonctions de compression et décompression utilisées par quelques programmes

6.12. File-5.13

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

Temps de construction estimé : 0.1 SBU
Espace disque requis : 12.5 Mo

6.12.1. 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
```

6.12.2. Contenu de File

Programmes installés: file
Bibliothèque installée: libmagic.{a,so}

Descriptions courtes

file Tente de classifier chaque fichier donné. Il réalise ceci en exécutant différents tests—tests sur le système de fichiers, tests des nombres magiques et tests de langages

`libmagic` Contient des routines pour la reconnaissance de nombres magiques utilisés par le programme **file**

6.13. Binutils-2.23.1

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

Temps de construction estimé : 1.9 SBU
Espace disque requis : 343 Mo

6.13.1. Installation de Binutils

Vérifiez que les pseudo-terminaux (PTY) fonctionnent correctement dans l'environnement en effectuant un simple test :

```
expect -c "spawn ls"
```

Cette commande devrait afficher ce qui suit :

```
spawn ls
```

Si, à la place, la sortie affiche le message ci-dessous, c'est que l'environnement n'est pas paramétré pour la bonne opération PTY. Vous devez résoudre ce problème avant de lancer les suites de test de Binutils et de GCC :

```
The system has no more ptys.
Ask your system administrator to create more.
```

Supprimez l'installation d'un fichier obsolète `standards.info` puisqu'un plus récent est installé plus tard dans les instructions pour Autoconf :

```
rm -fv etc/standards.info
sed -i.bak '/^INFO/s/standards.info //' etc/Makefile.in
```

Corrigez quelques scripts de la suite de tests pour que tous les tests fonctionnent :

```
patch -Np1 -i ../binutils-2.23.1-testsuite_fix-1.patch
```

La documentation de Binutils recommande de construire Binutils à l'extérieur du répertoire des sources dans un répertoire dédié :

```
mkdir -v ../binutils-build
cd ../binutils-build
```

Préparez la compilation de Binutils :

```
../binutils-2.23.1/configure --prefix=/usr --enable-shared
```

Remarque

Il existe un argument facultatif à **configure**, `--enable-lto`, qu'on peut utiliser pour permettre aux commandes **ar**, **nm** et **ranlib** d'accepter un paramètre a `--plugin`. C'est utilisé pour permettre à **gcc** de faire "de l'optimisation du temps de l'édition de liens" si elle est spécifiée. Aucun paquet de LFS ou de BLFS n'utilise cette possibilité.

Compilez le paquet :

```
make tooldir=/usr
```

Voici la signification des options de configure :

`tooldir=/usr`

Normalement, le répertoire `tooldir` (celui où seront placés les exécutables) est configuré avec `$(exec_prefix)/$(target_alias)`. Par exemple, les machines x86_64 l'étendront en `/usr/x86_64-unknown-linux-gnu`. Comme il s'agit d'un système personnalisé, nous n'avons pas besoin d'un répertoire spécifique à notre cible dans `/usr`. `$(exec_prefix)/$(target_alias)` serait utilisée si le système avait pour but une cross-compilation (par exemple, compiler un paquet sur une machine Intel qui génère du code pouvant être exécuté sur des machines PowerPC).

Important

La suite de tests de Binutils dans cette section est considérée comme critique. Ne la sautez sous aucun prétexte.

Testez les résultats :

```
make -k check
```

Installez le paquet :

```
make tooldir=/usr install
```

Installez le fichier d'en-tête `libiberty` requis par certains paquets :

```
cp -v ../binutils-2.23.1/include/libiberty.h /usr/include
```

6.13.2. Contenu de Binutils

Programmes installés:	addr2line, ar, as, c++filt, elfedit, gprof, ld, ld.bfd, nm, objcopy, objdump, ranlib, readelf, size, strings et strip
Bibliothèques installées:	libiberty.a, libbfd.{a,so} et libopcodes.{a,so}
Répertoire installé:	/usr/lib/ldscripts

Descriptions courtes

addr2line	Traduit les adresses de programme en noms de fichier et numéros de ligne ; suivant une adresse et le nom d'un exécutable, il utilise les informations de débogage disponibles dans l'exécutable pour déterminer le fichier source et le numéro de ligne associé à cette adresse
ar	Crée, modifie et extrait à partir d'archives
as	Un assembleur qui assemble la sortie de **gcc** en un fichier objet
c++filt	Utilisé par l'éditeur de liens pour récupérer les symboles C++ et Java, et pour empêcher les fonctions surchargées d'arrêter brutalement le programme
elfedit	Met à jour l'en-tête ELF des fichiers ELF
gprof	Affiche les données de profilage d'appels dans un graphe
ld	Un éditeur de liens combinant un certain nombre d'objets et de fichiers archives en un seul fichier, en déplaçant leur données et en regroupant les références de symboles
ld.bfd	Lien dur vers **ld**
nm	Liste les symboles disponibles dans un fichier objet

objcopy	Traduit un type de fichier objet en un autre
objdump	Affiche des informations sur le fichier objet donné, les options contrôlant les informations à afficher ; l'information affichée est surtout utile aux programmeurs qui travaillent sur les outils de compilation
ranlib	Génère un index du contenu d'une archive et le stocke dans l'archive ; l'index liste tous les symboles définis par les membres de l'archive qui sont des fichiers objet déplaçables
readelf	Affiche des informations sur les binaires du type ELF
size	Liste les tailles des sections et la taille totale pour les fichiers objets donnés
strings	Affiche, pour chaque fichier donné, la séquence de caractères affichables qui sont d'au moins la taille spécifiée (par défaut, 4) ; pour les fichiers objets, il affiche, par défaut, seulement les chaînes des sections d'initialisation et de chargement alors que pour les autres types de fichiers, il parcourt le fichier entier
strip	Supprime les symboles des fichiers objets
`libiberty`	Contient des routines utilisées par différents programmes GNU comme **getopt**, **obstack**, **strerror**, **strtol**, et **strtoul**
`libbfd`	Bibliothèque des descripteurs de fichiers binaires (*Binary File Descriptor*)
`libopcodes`	Une bibliothèque de gestion des opcodes—la « version lisible » des instructions du processeur ; elle est utilisée pour construire des outils comme **objdump**.

6.14. GMP-5.1.0

Le paquet GMP contient des bibliothèques de maths. Elles contiennent des fonctions utiles pour l'arithmétique à précision arbitraire.

Temps de construction estimé : 1.2 SBU
Espace disque requis : 50 Mo

6.14.1. Installation de GMP

Remarque

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 ...
```

Préparez la compilation de GMP :

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

Voici la signification des nouvelles options de configure :

`--enable-cxx`
Ce paramètre active le support pour C++

Compilez le paquet :

```
make
```

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
```

Assurez-vous que tous les 184 tests de la suite de tests réussissent. Vérifiez les résultats en lançant la commande suivante :

```
awk '/tests passed/{total+=$2} ; END{print total}' gmp-check-log
```

Installez le paquet :

```
make install
```

Si désiré, installez la documentation :

```
mkdir -v /usr/share/doc/gmp-5.1.0
cp    -v doc/{isa_abi_headache,configuration} doc/*.html \
         /usr/share/doc/gmp-5.1.0
```

6.14.2. Contenu de GMP

Bibliothèques installées: libgmp.{a,so}, libgmpxx.{a,so}, and libmp.{a,so}
Répertoire installé: /usr/share/doc/gmp-5.1.0

Descriptions courtes

`libgmp`	Contient les fonctions de maths de précision.
`libgmpxx`	Contient des fonctions de maths de précision pour C++
`libmp`	Contient des fonctions de maths pour Berkeley MP.

6.15. MPFR-3.1.1

Le paquet MPFR contient des fonctions pour des maths à précision multiple.

Temps de construction estimé : 0.8 SBU
Espace disque requis : 27 Mo

6.15.1. Installation de MPFR

Préparez la compilation de MPFR :

```
./configure  --prefix=/usr        \
             --enable-thread-safe \
             --docdir=/usr/share/doc/mpfr-3.1.1
```

Compilez le paquet :

```
make
```

Important

La suite de tests de MPFR est considérée comme critique. Ne la sautez en aucun cas.

Testez les résultats et assurez-vous que tous les tests ont réussi :

```
make check
```

Installez le paquet :

```
make install
```

Installez la documentation :

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

6.15.2. Contenu de MPFR

Bibliothèques installées: libmpfr.{a,so}
Répertoire installé: /usr/share/doc/mpfr-3.1.1

Descriptions courtes

`libmpfr` Contient des fonctions de maths à précision multiple.

6.16. MPC-1.0.1

Le paquet MPC contient une bibliothèque pour le calcul arithmétique de nombres complexes avec une haute précision au choix et l'arrondissement correcte du résultat.

Temps de construction estimé : 0.4 SBU
Espace disque requis : 10.2 Mo

6.16.1. Installation de MPC

Préparez la compilation de MPC :

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

Compilez le paquet :

```
make
```

Pour tester les résultats, lancez :

```
make check
```

Installez le paquet :

```
make install
```

6.16.2. Contenu de MPC

Bibliothèques installées: libmpc.{a,so}

Descriptions courtes

`libmpc` Contient des fonctions mathématiques complexes

6.17. GCC-4.7.2

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

Temps de construction estimé : 53.5 SBU
Espace disque requis : 2.0 Go

6.17.1. Installation de GCC

Appliquez une substitution **sed** qui supprimera l'installation de `libiberty.a`. À la place, la version de `libiberty.a` fournie par Binutils sera utilisée :

```
sed -i 's/install_to_$(INSTALL_DEST) //' libiberty/Makefile.in
```

Là encore, ne construisez pas les fichiers .info. Ils ne sont pas nécessaires ici et ils sont cassés dans la version actuelle de **makeinfo**.

```
sed -i 's/BUILD_INFO=info/BUILD_INFO=/' gcc/configure
```

Comme au Section 5.9, « GCC-4.7.2 - Passe 2 », appliquez la commande **sed** suivante pour obliger la construction à utiliser le drapeau de construction `-fomit-frame-pointer` afin de garantir des constructions de compilateur cohérentes :

```
case `uname -m` in
  i?86) sed -i 's/^T_CFLAGS =$/& -fomit-frame-pointer/' gcc/Makefile.in ;;
esac
```

Corrigez aussi une erreur dans un des Makefiles de vérification :

```
sed -i -e /autogen/d -e /check.sh/d fixincludes/Makefile.in
```

La documentation de GCC recommande de construire GCC en dehors du répertoire source, c'est-à-dire dans un répertoire dédié :

```
mkdir -v ../gcc-build
cd ../gcc-build
```

Préparez la compilation de GCC :

```
../gcc-4.7.2/configure --prefix=/usr            \
                       --libexecdir=/usr/lib    \
                       --enable-shared          \
                       --enable-threads=posix   \
                       --enable-__cxa_atexit    \
                       --enable-clocale=gnu     \
                       --enable-languages=c,c++ \
                       --disable-multilib       \
                       --disable-bootstrap      \
                       --with-system-zlib
```

Remarquez que pour d'autres langages, il existe des prérequis non disponibles. Voir le livre BLFS pour des instructions sur la manière de construire tous les langages supportés par GCC.

Voici la signification de la nouvelle option de configure :

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

Remarquez que pour d'autres langages, il y a des prérequis qui ne sont pas disponibles. Voir le livre BLFS pour des instructions sur la façon de construire tous les langages supportés par GCC.

Remarque

Il existe un argument facultatif à **configure**, `--enable-lto`, qu'on peut utiliser pour permettre à la commande **gcc** de faire "de l'optimisation du temps de l'édition de liens" si elle est spécifiée. Aucun paquet de LFS ou de BLFS n'utilise cette possibilité.

Pour utiliser cette fonctionnalité, elle doit également être activée dans binutils.

Compilez le paquet :

```
make
```

Important

Dans cette section, la suite de tests pour GCC est considérée comme critique. Ne les sautez sous aucun prétexte.

Un ensemble de tests dans la suite de tests de GCC est connu pour déborder la pile, donc augmentez la taille de la pile avant de lancer les tests :

```
ulimit -s 32768
```

Testez les résultats mais ne vous arrêtez pas aux erreurs :

```
make -k check
```

Pour recevoir un résumé des résultats de la suite de tests, lancez

```
../gcc-4.7.2/contrib/test_summary
```

Pour n'avoir que les résumés, redirigez la sortie vers **`grep -A7 Summ`**.

Vous pouvez comparer les résultats avec ceux situés dans *http://www.linuxfromscratch.org/lfs/build-logs/7.3/* et *http://gcc.gnu.org/ml/gcc-testresults/*.

Quelques échecs inattendus sont inévitables. Les développeurs de GCC connaissent ces problèmes, mais ne les ont pas encore résolus. En particulier, les tests de `libmudflap` sont connus pour être particulièrement problématiques et résultant d'un bogue dans GCC (*http://gcc.gnu.org/bugzilla/show_bug.cgi?id=20003*). Sauf si les résultats du test sont très différents de ceux sur l'adresse ci-dessus, vous pouvez continuer en toute sécurité.

Installez le paquet :

```
make install
```

Quelques paquets s'attendent à ce que le préprocesseur C soit installé dans le répertoire `/lib` Pour supporter ces paquets, créez ce lien symbolique :

```
ln -sv ../usr/bin/cpp /lib
```

Beaucoup de paquets utilisent le nom **cc** pour appeler le compilateur C. Pour satisfaire ces paquets, créez un lien symbolique :

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

Maintenant que notre chaîne d'outils est en place, il est important de s'assurer à nouveau que la compilation et l'édition de liens fonctionneront comme prévu. Cela se fait en effectuant les mêmes tests de propreté que ceux faits plus haut dans ce chapitre :

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

Si tout fonctionne correctement, il ne devrait pas y avoir d'erreur et la sortie de la commande sera (avec des différences spécifiques aux plateformes dans le nom de l'éditeur de liens) :

```
[Requesting program interpreter: /lib/ld-linux.so.2]
```

Maintenant, assurez-vous que nous utilisons les bons fichiers de démarrage :

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

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

```
/usr/lib/gcc/i686-pc-linux-gnu/4.7.2/../../../crt1.o succeeded
/usr/lib/gcc/i686-pc-linux-gnu/4.7.2/../../../crti.o succeeded
/usr/lib/gcc/i686-pc-linux-gnu/4.7.2/../../../crtn.o succeeded
```

Selon l'architecture de votre machine, le message ci-dessus peut légèrement différer, la différence portant normalement sur le nom du répertoire après `/usr/lib/gcc`. Si votre machine est un système 64 bits, il se peut que vous voyiez un répertoire nommé `lib64` vers la fin de la chaîne. La chose importante à chercher est que **gcc** ait trouvé les trois `crt*.o` sous le répertoire `/usr/lib`.

Vérifiez que le compilateur cherche les bons fichiers d'en-tête :

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

Cette commande devrait réussir avec la sortie suivante :

```
#include <...> search starts here:
 /usr/local/include
 /usr/lib/gcc/i686-pc-linux-gnu/4.7.2/include
 /usr/lib/gcc/i686-pc-linux-gnu/4.7.2/include-fixed
 /usr/include
```

A nouveau, notez que le nom du répertoire après votre triplette cible peut être différent de celui ci-dessus, selon votre architecture.

Remarque

Depuis la version 4.3.0, GCC installe maintenant sans condition le fichier `limits.h` dans un répertoire à part `include-fixed`, et ce répertoire doit être en place.

Puis, vérifiez que le nouvel éditeur de liens est utilisé avec les bons chemins de recherche :

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

Si tout fonctionne correctement, il ne devrait pas y avoir d'erreur et la sortie de la dernière commande sera (selon la triplette cible spécifique à chaque plateforme) :

```
SEARCH_DIR("/usr/i686-pc-linux-gnu/lib")
SEARCH_DIR("/usr/local/lib")
SEARCH_DIR("/lib")
SEARCH_DIR("/usr/lib");
```

Il se peut qu'un système 64 bits voie un peu plus de répertoires. Par exemple, voici la sortie d'une machine x86_64 :

```
SEARCH_DIR("/usr/x86_64-unknown-linux-gnu/lib64")
SEARCH_DIR("/usr/local/lib64")
SEARCH_DIR("/lib64")
SEARCH_DIR("/usr/lib64")
SEARCH_DIR("/usr/x86_64-unknown-linux-gnu/lib")
SEARCH_DIR("/usr/local/lib")
SEARCH_DIR("/lib")
SEARCH_DIR("/usr/lib");
```

Ensuite, assurez-vous que nous utilisons la bonne libc :

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

Si tout fonctionne correctement, il ne devrait pas y avoir d'erreur et la sortie de la dernière commande sera (selon la triplette cible spécifique à chaque plateforme) :

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

Pour finir, assurez-vous que GCC utilise le bon éditeur de liens dynamiques :

```
grep found dummy.log
```

Si tout fonctionne correctement, il ne devrait pas y avoir d'erreur et la sortie de la commande sera (avec des différences spécifiques aux plateformes dans le nom de l'éditeur de liens et un répertoire lib64 sur les hôtes 64 bits) :

```
found ld-linux.so.2 at /lib/ld-linux.so.2
```

Si la sortie n'apparaît pas comme montré ci-dessus ou qu'elle n'apparaît pas du tout, alors quelque chose ne va vraiment pas. Enquêtez et retracez les étapes pour savoir d'où vient le problème et comment le corriger. La raison la plus probable est que quelque chose s'est mal passé lors de la modification du fichier specs ci-dessus. Tout problème devra être résolu avant de continuer le processus.

Une fois que tout fonctionne correctement, nettoyez les fichiers tests :

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

Enfin, déplacez un fichier mal placé :

```
mkdir -pv /usr/share/gdb/auto-load/usr/lib
mv -v /usr/lib/*gdb.py /usr/share/gdb/auto-load/usr/lib
```

6.17.2. Contenu de GCC

Programmes installés: c++, cc (lien vers gcc), cpp, g++, gcc, gcc-ar, gcc-nm, gcc-ranlib, gccbug, et gcov
Bibliothèques installés: libgcc.a, libgcc_eh.a, libgcc_s.so, libgcov.a, libgomp.{a,so}, liblto_plugin.so, libmudflap.{a,so}, libmudflapth.{a,so}, libquadmath.{a,so}, libssp.{a,so}, libssp_nonshared.a, libstdc++.{a,so} et libsupc++.a
Répertoires installés: /usr/include/c++, /usr/lib/gcc, /usr/share/gcc-4.7.2

Descriptions courtes

c++ Le compilateur C++

cc Le compilateur C

cpp Le préprocesseur C ; il est utilisé par le compilateur pour l'extension des instructions #include, #define et d'autres instructions similaires dans les fichiers sources

g++ Le compilateur C++

gcc Le compilateur C

gcc-ar Une enveloppe autour de **ar** qui ajoute un greffon à la ligne de commande. Ce programme n'est utilisé que pour ajouter "l'optimisation du temps d'édition des liens" et il n'est pas utile avec les options de construction par défaut.

gcc-nm Une enveloppe autour de **nm** qui ajoute un greffon à la ligne de commande. Ce programme n'est utilisé que pour ajouter "l'optimisation du temps d'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 du temps d'édition des liens" et il n'est pas utile avec les options de construction par défaut.

gccbug Un script shell utilisé pour aider à la création de bons rapports de bogues

gcov Un outil de tests ; il est utilisé pour analyser les programmes et savoir où des optimisations seraient suivies du plus d'effet

`libgcc` Contient un support en exécution pour **gcc**

`libgcov` Cette bibliothèque est liée à un programme où on demande à GCC d'activer le profiling

`libgomp` Implémentation GNU de l'API OpenMP API pour la programmation en mémoire parallèle partagée pour plusieurs plateforme en C/C++ et Fortran

`liblto_plugin` plugin *GCC's Link Time Optimization* (LTO, optimisation du temps d'édition de liens de GCC) permettant à GCC de pratiquer des optimisations tout au cours des unités de compilation.

`libmudflap` Contient des routines qui supportent la fonctionnalité de test des limites de GCC

`libquadmath` API de la bibliothèque de maths GCC de précision au carré

`libssp` Contient des routines supportant la fonctionalité de GCC de protection contre les débordements de mémoire

`libstdc++`	La bibliothèque C++ standard
`libsupc++`	Fournit des routines de support pour le langage de programmation C++

6.18. Sed-4.2.2

Le paquet Sed contient un éditeur de flux.

Temps de construction estimé : 0.2 SBU
Espace disque requis : 6.7 Mo

6.18.1. Installation de Sed

Préparez la compilation de Sed :

```
./configure --prefix=/usr --bindir=/bin --htmldir=/usr/share/doc/sed-4.2.2
```

Voici la signification des options de configuration :

`--htmldir`
Ceci indique le répertoire où la documentation HTML sera installée.

Compilez le paquet :

```
make
```

Générez la documentation HTML :

```
make html
```

Pour tester les résultats, lancez :

```
make check
```

Installez le paquet :

```
make install
```

Installez la documentation HTML :

```
make -C doc install-html
```

6.18.2. Contenu de Sed

Programme installé: sed
Répertoire installé: /usr/share/doc/sed-4.2.2

Description courte

sed Filtre et transforme des fichiers texte en une seule passe

6.19. Bzip2-1.0.6

Le paquet Bzip2 contient des programmes de compression et décompression de fichiers. Compresser des fichiers texte avec **bzip2** permet d'atteindre un taux de compression bien meilleur qu'avec l'outil **gzip**.

Temps de construction estimé : moins de 0.1 SBU

Espace disque requis : 6.9 Mo

6.19.1. Installation de Bzip2

Appliquez un correctif qui installera la documentation de ce paquet :

```
patch -Np1 -i ../bzip2-1.0.6-install_docs-1.patch
```

La commande suivante garantit l'installation de liens symboliques relatifs :

```
sed -i 's@\(ln -s -f \)$(PREFIX)/bin/@\1@' Makefile
```

Assurez-vous que les pages de manuel s'installent au bon endroit :

```
sed -i "s@(PREFIX)/man@(PREFIX)/share/man@g" Makefile
```

Préparez la compilation de Bzip2 avec :

```
make -f Makefile-libbz2_so
make clean
```

Voici la signification du paramètre de make :

`-f Makefile-libbz2_so`

Ceci fera que Bzip2 sera construit en utilisant un fichier `makefile` différent, dans ce cas le fichier `Makefile-libbz2_so` qui crée une bibliothèque `libbz2.so` dynamique et lie les outils Bzip2 avec.

Compilez et testez le paquet :

```
make
```

Installez les programmes :

```
make PREFIX=/usr install
```

Installez le binaire dynamique **bzip2** dans le répertoire `/bin`, créez les liens symboliques nécessaires et nettoyez :

```
cp -v bzip2-shared /bin/bzip2
cp -av libbz2.so* /lib
ln -sv ../../lib/libbz2.so.1.0 /usr/lib/libbz2.so
rm -v /usr/bin/{bunzip2,bzcat,bzip2}
ln -sv bzip2 /bin/bunzip2
ln -sv bzip2 /bin/bzcat
```

6.19.2. 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.{a,so}
Répertoire installé: /usr/share/doc/bzip2-1.0.6

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

6.20. Pkg-config-0.27.1

Le paquet pkg-config contient un outil pour passer le chemin include et/ou les chemins des bibliothèques afin de construire les outils au moment de l'exécution de configure et de make file.

Temps de construction estimé : 0.4 SBU
Espace disque requis : 30 Mo

6.20.1. Installation de Pkg-config

Préparez la compilation de Pkg-config :

```
./configure --prefix=/usr         \
            --with-internal-glib  \
            --docdir=/usr/share/doc/pkg-config-0.27.1
```

Compilez le paquet :

```
make
```

Pour tester les résultats, lancez

```
make check
```

Installez le paquet :

```
make install
```

6.20.2. Contenu de Pkg-config

Programme installé: pkg-config
Répertoire installé: /usr/share/doc/pkg-config-0.27.1

Descriptions courtes

pkg-config retourne des méta informations sur la bibliothèque ou le paquet spécifié.

6.21. Ncurses-5.9

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

Temps de construction estimé : 0.6 SBU

Espace disque requis : 40 Mo

6.21.1. Installation de Ncurses

Préparez la compilation de Ncurses :

```
./configure --prefix=/usr           \
            --mandir=/usr/share/man \
            --with-shared           \
            --without-debug         \
            --enable-pc-files       \
            --enable-widec
```

Voici la signification des options de configure :

`--enable-widec`
Cette option amène les bibliothèques « wide-character » (comme `libncursesw.so.5.9`) à être compilée au lieu de celles normales (comme `libncurses.so.5.9`). 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.

Compilez le paquet :

```
make
```

Ce paquet a une suite de tests, mais elle ne peut être exécutée qu'après que le paquet a été installé. Les tests se situent dans le répertoire `test/`. Voir le fichier `README` dans ce répertoire pour de plus amples détails.

Installez le paquet :

```
make install
```

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

```
mv -v /usr/lib/libncursesw.so.5* /lib
```

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

```
ln -sfv ../../lib/libncursesw.so.5 /usr/lib/libncursesw.so
```

Beaucoup d'applications s'attendent encore à ce que l'éditeur de liens puisse trouver les bibliothèques Ncurses non wide-character. Faites croire à de telles applications au lien vers les bibliothèques « with wide-character » par des faux liens symboliques et des scripts d'éditeur de liens :

```
for lib in ncurses form panel menu ; do
    rm -vf                    /usr/lib/lib${lib}.so
    echo "INPUT(-l${lib}w)" > /usr/lib/lib${lib}.so
    ln -sfv lib${lib}w.a      /usr/lib/lib${lib}.a
    ln -sfv ${lib}w.pc        /usr/lib/pkgconfig/${lib}.pc
    ln -sfv lib${lib}w.a /usr/lib/lib${lib}.a
done
ln -sfv libncurses++w.a /usr/lib/libncurses++.a
```

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
ln -sfv libncursesw.a      /usr/lib/libcursesw.a
ln -sfv libncurses.a       /usr/lib/libcurses.a
```

Si désiré, installez la documentation de Ncurses :

```
mkdir -v       /usr/share/doc/ncurses-5.9
cp -v -R doc/* /usr/share/doc/ncurses-5.9
```

Remarque

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. Si vous devez avoir de telles bibliothèques à cause d'une application disponible qu'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
make sources libs
cp -av lib/lib*.so.5* /usr/lib
```

6.21.2. Contenu de Ncurses

Programmes installés: captoinfo (lien vers tic), clear, infocmp, infotocap (lien vers tic), ncursesw5-config, reset (lien vers tset), tabs, tic, toe, tput et tset

Bibliothèques installées: libcursesw.{a,so} (lien symbolique et script de l'éditeur de liens vers libncursesw.{a,so}), libformw.{a,so}, libmenuw.{a,so}, libncurses++w.a, libncursesw.{a,so}, libpanelw.{a,so} ainsi que leurs équivalents non « wide-character » avec un nom identique, mais sans le w.

Répertoires installés: /usr/share/tabset, /usr/share/terminfo, /usr/share/doc/ncurses-5.9

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
ncursesw5-config	Fournit des informations de configuration de ncurses
reset	Réinitialise un terminal avec ses valeurs par défaut
tabs	Efface et initialise des taquets de tab sur un terminal
tic	Le compilateur d'entrée de description terminfo, traduisant un fichier terminfo au format source dans un format binaire nécessaire pour les routines des bibliothèques ncurses. Un fichier terminfo contient des informations sur les capacités d'un terminal particulier
toe	Liste tous les types de terminaux disponibles, donnant pour chacun d'entre eux son nom principal et sa description
tput	Rend les valeurs de capacités dépendant du terminal disponibles au shell ; il peut aussi être utilisé pour réinitialiser un terminal ou pour afficher son nom long
tset	Peut être utilisé pour initialiser des terminaux
`libcurses`	Un lien vers `libncurses`
`libncurses`	Contient des fonctions pour afficher du texte de plusieurs façons compliquées sur un écran de terminal ; un bon exemple d'utilisation de ces fonctions est le menu affiché par le **make menuconfig** du noyau
`libform`	Contient des fonctions pour implémenter des formes
`libmenu`	Contient des fonctions pour implémenter des menus
`libpanel`	Contient des fonctions pour implémenter des panneaux

6.22. Util-linux-2.22.2

Le paquet Util-linux contient différents outils. Parmi eux se trouvent des outils de gestion des systèmes de fichiers, de consoles, de partitions et des messages.

Temps de construction estimé : 0.7 SBU
Espace disque requis : 81 Mo

6.22.1. Notes de compatibilité FHS

Le FHS recommande d'utiliser le répertoire `/var/lib/hwclock` au lieu de l'habituel `/etc` comme emplacement du fichier `adjtime`. Pour rendre **hwclock** compatible avec le FHS, lancez ce qui suit :

```
sed -i -e 's@etc/adjtime@var/lib/hwclock/adjtime@g' \
     $(grep -rl '/etc/adjtime' .)
mkdir -pv /var/lib/hwclock
```

6.22.2. Installation d'Util-linux

```
./configure --disable-su --disable-sulogin --disable-login
```

Voici la signification de l'option de configure :

`--disable-*`
Ces paramètres désactivent la construction de su, sulogin et de login. Ils sont des doublons des programmes fournis par Section 6.26, « Shadow-4.1.5.1 » et Section 6.58, « Sysvinit-2.88dsf ». Ils exigent par ailleurs *Linux-PAM* qui n'est pas disponible dans LFS.

Compilez le paquet :

```
make
```

Ce paquet n'est fourni avec aucune suite de tests.

Installez le paquet :

```
make install
```

6.22.3. Contenu d'Util-linux

Programmes installés: addpart, agetty, blkid, blockdev, cal, cfdisk, chcpu, chrt, col, colcrt, colrm, column, ctrlaltdel, cytune, delpart, dmesg, eject, fallocate, fdformat, fdisk, findfs, findmnt, flock, fsck, fsck.cramfs, fsck.minix, fsfreeze, getopt, hexdump, hwclock, i386, ionice, ipcmk, ipcrm, ipcs, isosize, ldattach, linux32, linux64, logger, look, losetup, lsblk, lscpu, lslocks, mcookie, mkfs, mkfs.bfs, mkfs.cramfs, mkfs.minix, mkswap, more, mount, mountpoint, namei, partx, pg, pivot_root, prlimit, raw, readprofile, rename, renice, resizepart, rev, rtcwake, script, scriptreplay, setarch, setsid, setterm, sfdisk, swaplabel, swapoff (link to swapon), swapon, switch_root, tailf, taskset, tunelp, ul, umount, unshare, utmpdump, uuidd, uuidgen, wall, wdctl, whereis, wipefs et x86_64
Bibliothèques installées: libblkid.{a,so}, libmount.{a,so}, libuuid.{a,so}
Répertoires installés: /usr/include/blkid, /usr/include/libmount, /usr/include/uuid, /usr/share/getopt, /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**
blkid	Un outil en ligne de commande pour trouver et afficher les attributs d'un périphérique bloc
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
chrt	Manipule les attributs d'un processus en temps réel
col	Filtre les retours 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
cytune	Est utilisé pour paramétrer finement les pilotes de lignes séries des cartes Cyclades
delpart	Demande au noyau Linux de supprimer une partition
dmesg	Affiche les messages du noyau lors du démarrage
eject	Éjecte un média amovible
fallocate	Pré-alloue de l'espace à un fichier
fdformat	Réalise un formatage de bas niveau sur un disque amovible
fdisk	Est utilisé pour manipuler la table de partitions du périphérique donné
findfs	Trouve un système de fichiers par label 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 matériel, aussi appelée horloge RTC (*Real-Time Clock*, horloge à temps réel) ou horloge BIOS (*Basic Input-Output System*)

i386	Un lien symbolique vers setarch
ionice	Obtient ou initialise la classe de planification IO (ES) et la priorité pour un programme
ipcmk	Crée diverses ressources IPC
ipcrm	Supprime la ressource IPC (inter-process communication) donnée
ipcs	Fournit l'information de statut IPC
isosize	Affiche la taille d'un système de fichiers iso9660
kill	Envoie des signaux aux processus
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 avec 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
lslocks	Liste les verrous du système local
mcookie	Génère des cookies magiques, nombres hexadécimaux aléatoires sur 128 bits, pour **xauth**
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
partx	Signale au noyau la présence et le nombre de partitions sur un disque
pg	Affiche un fichier texte un écran à la fois
pivot_root	Fait en sorte que le système de fichiers donné soit le nouveau système de fichiers racine du processus actuel
prlimit	Récupère et envoie la limite des ressources d'un processus
raw	Envoie un périphérique de caractère de base Linux vers un périphérique de bloc
readprofile	Lit les informations de profilage du noyau
rename	Renomme les fichiers donnés, remplaçant une chaîne donnée par une autre
renice	Modifie la priorité des processus exécutés

resizepart Demande au noyau Linux de redimensionner une partition

rev Inverse les lignes d'un fichier donné

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, de tout ce qui est affiché sur un terminal

scriptreplay Rejoue des scripts type en utilisant les informations de temps

setarch Change d'architecture signalée dans un nouvel environnement de programme et initialise les commutateurs adéquats

setsid Lance le programme donné dans une nouvelle session

setterm Initialise les attributs du terminal

sfdisk Est un manipulateur de table de partitions disque

swaplabel Permet de modifier l'UUID et l'étiquette d'un espace d'échange

swapoff Désactive des périphériques et des fichiers pour la pagination et l'échange

swapon Active les périphériques et fichiers de pagination et d'échange et liste les périphériques et fichiers en cours d'utilisation.

switch_root Change de système de fichiers racine pour une arborescence montée

tailf Observe la croissance d'un fichier journal. Affiche les 10 dernières lignes d'un fichier journal, puis continue à afficher toute nouvelle entrée dans le fichier journal dès qu'elle est créée

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

tunelp Est utilisé pour paramétrer finement une imprimante ligne

ul Un filtre pour traduire les souslignements en séquences d'échappement indiquant un souslignement pour le terminal utilisé

umount Déconnecte un système de fichiers à partir de la hiérarchie de fichiers du système

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 unique.

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.

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 watchdog du 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

`libblkid` Contient des routines pour l'identification des périphériques et l'extraction des modèles

`libmount` Contient les routines pour le montage et le démontage des périphériques de bloc

`libuuid` Contient des routines pour la génération d'identifiants uniques pour des objets qui peuvent être accessibles en-dehors du système local

6.23. Psmisc-22.20

Le paquet Psmisc contient des programmes pour afficher des informations sur les processus en cours d'exécution.

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

6.23.1. Installation de Psmisc

Préparez la compilation de Psmisc pour :

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

Compilez le paquet :

```
make
```

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

Installez le paquet :

```
make install
```

Enfin, déplacez les programmes **killall** et **fuser** à l'endroit spécifié par la FHS :

```
mv -v /usr/bin/fuser   /bin
mv -v /usr/bin/killall /bin
```

6.23.2. Contenu de Psmisc

Programmes installés: fuser, killall, peekfd, prtstat, 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
pstree	Affiche les processus en cours hiérarchiquement
pstree.x11	Identique à **pstree**, si ce n'est qu'il attend une confirmation avant de quitter

6.24. Procps-3.3.6

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

Temps de construction estimé : 0.2 SBU
Espace disque requis : 13 Mo

6.24.1. Installation de Procps-ng

Préparez maintenant la compilation de procps-ng :

```
./configure --prefix=/usr                            \
            --exec-prefix=                           \
            --libdir=/usr/lib                        \
            --docdir=/usr/share/doc/procps-ng-3.3.6 \
            --disable-static                         \
            --disable-skill                          \
            --disable-kill
```

Voici la signification des options de configure :

`--disable-skill`
Ce paramètre désactive les commandes obsolètes et non portables skill et snice.

`--disable-kill`
Cette option désactive la construction de la commande kill installée dans le paquet util-linux.

Compilez le paquet :

```
make
```

La suite de tests exige des modifications pour LFS. La commande **which** n'est pas disponible et le test **pmap** n'associe pas de caractère newline pour deux tests, par ailleurs il se peut que le test **slabtop** renvoie 999,999 objets. Pour lancer la suite de tests, lancez les commandes suivantes :

```
pushd testsuite
  sed -i -e 's|exec which sleep|exec echo /tools/bin/sleep|' \
           -e 's|999999|&9|'    config/unix.exp
sed -i -e 's|pmap_initname\\\$|pmap_initname|' pmap.test/pmap.exp
  sed -i 's|pmap_initname\\\$|pmap_initname|' pmap.test/pmap.exp
  make site.exp
  DEJAGNU=global-conf.exp runtest
popd
```

Installez le paquet :

```
make install
```

Enfin, déplacez la bibliothèque à un endroit où elle sera trouvable si `/usr` n'est pas monté.

```
mv -v /usr/lib/libprocps.so.* /lib
ln -sfv ../../lib/libprocps.so.1.0.0 /usr/lib/libprocps.so
```

6.24.2. Contenu de Procps-ng

Programmes installés: free, pgrep, pkill, pmap, ps, pwdx, slabtop, sysctl, tload, top, uptime, vmstat, w, et watch
Bibliothèque installée: libprocps.so

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

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

pwdx Indique le répertoire d'exécution courant d'un processus

slabtop Affiche des informations détaillées sur le cache slap 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` Contient les fonctions utilisées par la plupart des programmes de ce paquet

6.25. E2fsprogs-1.42.7

Le paquet E2fsprogs contient les outils de gestion du système de fichiers `ext2`. Il supporte aussi les systèmes de fichiers journalisés `ext3` et `ext4`.

Temps de construction estimé : 1.7 SBU
Espace disque requis : 64 Mo

6.25.1. Installation de E2fsprogs

Il est recommandé par la documentation de construire E2fsprogs dans un sous-répertoire du répertoire source :

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

Préparez la compilation d'E2fsprogs :

```
../configure --prefix=/usr         \
             --with-root-prefix="" \
             --enable-elf-shlibs   \
             --disable-libblkid    \
             --disable-libuuid     \
             --disable-uuidd       \
             --disable-fsck
```

Voici la signification des options de configure :

`--with-root-prefix=""`
Certains programmes (comme **e2fsck** sont considérés comme essentiels. Quand, par exemple, `/usr` n'est pas monté, ces programmes essentiels doivent encore être disponibles. Ils appartiennent aux répertoires comme `/lib` et `/sbin`. Si cette option n'est pas passée au configure d'E2fsprogs, les programmes sont placés dans le répertoire `/usr`.

`--enable-elf-shlibs`
Ceci crée les bibliothèques partagées que certains programmes de ce paquet utilisent.

`--disable-*`
Ceci empêche E2fsprogs de construire et d'installer les bibliothèques `libuuid` et `libblkid`, le démon `uuidd` et l'emballeur **fsck**, qui ont été installés plus haut par Util-Linux.

Compilez le paquet :

```
make
```

Pour tester les résultats, lancez :

```
make -k check
```

Un des tests de E2fsprogs tentera d'allouer 256 Mo de mémoire. Si vous n'avez guère plus de RAM, il est recommandé d'activer un espace swap suffisant pour le test. Voir Section 2.3, « Créer un système de fichiers sur la partition » et Section 2.4, « Monter la nouvelle partition » pour des détails sur la création et l'activation de l'espace swap.

Installez les binaires et la documentation :

```
make install
```

Installez les bibliothèques statiques et les en-têtes :

```
make install-libs
```

Autorisez l'écriture dans les bibliothèques statiques installées pour que les symboles de débogage puissent être supprimés plus tard :

```
chmod -v u+w /usr/lib/{libcom_err,libe2p,libext2fs,libss}.a
```

Ce paquet installe le 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
```

6.25.2. Contenu de E2fsprogs

Programmes installés:	badblocks, chattr, compile_et, debugfs, dumpe2fs, e2freefrag, e2fsck, e2image, e2initrd_helper, e2label, e2undo, e4defrag, filefrag, fsck.ext2, fsck.ext3, fsck.ext4, fsck.ext4dev, logsave, lsattr, mk_cmds, mke2fs, mkfs.ext2, mkfs.ext3, mkfs.ext4, mkfs.ext4dev, mklost+found, resize2fs et tune2fs
Bibliothèques installées:	libcom_err.{a,so}, libe2p.{a,so}, libext2fs.{a,so}, libquota.a and libss.{a,so}
Répertoire installé:	/usr/include/e2p, /usr/include/et, /usr/include/ext2fs, /usr/include/quota, /usr/include/ss, /usr/share/et, /usr/share/ss

Descriptions courtes

badblocks	Recherche les blocs défectueux sur un périphérique (habituellement une partition d'un disque)
chattr	Modifie les attributs de fichiers sur un système de fichiers `ext2` et `ext3`, la version journalisée d'`ext2`
compile_et	Un compilateur de table d'erreurs. Il convertit une table de noms d'erreurs et des messages associés en un fichier source C à utiliser avec la bibliothèque `com_err`
debugfs	Un débogueur de système de fichiers. Il est utilisé pour examiner et modifier l'état d'un système de fichiers `ext2`
dumpe2fs	Affiche le superbloc et les informations de groupes de blocs sur le système de fichiers présent sur un périphérique donné
e2freefrag	Donne des informations de défragmentation de l'espace libre
e2fsck	Est utilisé pour vérifier, et quelque fois réparer, les systèmes de fichiers `ext2` et `ext3`

e2image	Est utilisé pour sauver les données critiques d'un système de fichiers `ext2` dans un fichier
e2initrd_helper	Affiche le type de système de fichiers, d'un nom de périphérique ou d'une étiquette donnés
e2label	Affiche ou modifie le label d'un système de fichiers `ext2` présent sur un périphérique donné
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.
e4defrag	Défragmenteur en ligne des systèmes de fichiers ext4
filefrag	Renseigne sur le niveau de fragmentation que peut atteindre un fichier
fsck.ext2	Vérifie par défaut les systèmes de fichiers `ext2`. C'est un lien vers **e2fsck**.
fsck.ext3	Vérifie par défaut les systèmes de fichiers `ext3`. C'est un lien vers **e2fsck**.
fsck.ext4	Vérifie par défaut les systèmes de fichiers `ext4`. C'est un lien vers **e2fsck**.
fsck.ext4dev	Vérifie par défaut les systèmes de fichiers de développement `ext3`. C'est un lien vers **e2fsck**.
logsave	Sauvegarde la sortie d'une commande dans un journal applicatif
lsattr	Liste les attributs de fichiers sur un système de fichiers ext2 (second extended file system)
mk_cmds	Convertit une table de noms de commandes et de messages d'aide en un fichier source C bon à utiliser avec la bibliothèque sous-système `libss`
mke2fs	Crée un système de fichiers `ext2` ou `ext3` sur le périphérique donné
mkfs.ext2	Crée par défaut un système de fichiers `ext2`. C'est un lien vers **mke2fs**.
mkfs.ext3	Crée par défaut un système de fichiers `ext3`. C'est un lien vers **mke2fs**.
mkfs.ext4	Crée par défaut un système de fichiers `ext4`. C'est un lien vers **mke2fs**.
mkfs.ext4dev	Crée par défaut les systèmes de fichiers de développement `ext4`. C'est un lien vers **fsck**.
mklost+found	Est utilisé pour créer un répertoire `lost+found` sur un système de fichiers `ext2` ; il pré-alloue des blocs disque dans ce répertoire pour faciliter la tâche d'**e2fsck**
resize2fs	Utilisé pour agrandir ou réduire un système de fichiers `ext2`
tune2fs	Ajuste les paramètres d'un système de fichiers `ext2`
`libcom_err`	La routine d'affichage d'erreurs
`libe2p`	Est utilisé par **dumpe2fs**, **chattr**, et **lsattr**
`libext2fs`	Contient des routines pour permettre aux programmes niveau utilisateur de manipuler un système de fichiers `ext2`
`libquota`	Offre une interface pour créer et mettre à jour des fichiers de quota et des champs de superblocs ext4
`libss`	Est utilisé par **debugfs**

6.26. Shadow-4.1.5.1

Le paquet Shadow contient des programmes de gestion de mots de passe d'une façon sécurisée.

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

6.26.1. Installation de Shadow

Remarque

Si vous aimeriez multiplier l'usage des mots de passe efficaces, reportez-vous à *http://lfs.traduc.org/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 sa page man car Coreutils fournit une meilleure version :

```
sed -i 's/groups$(EXEEXT) //' src/Makefile.in
find man -name Makefile.in -exec sed -i 's/groups\.1 / /' {} \;
```

Au lieu d'utiliser la méthode *crypt* par défaut, utilisez la méthode *SHA-512* plus sécurisée du chiffrement de mot de passe, qui autorise aussi les mots de passe plus longs que huit caractères. Il est également nécessaire de changer l'endroit obsolète de `/var/spool/mail` pour les boîtes e-mail de l'utilisateur que Shadow utilise par défaut en l'endroit `/var/mail` utilisé actuellement :

```
sed -i -e 's@#ENCRYPT_METHOD DES@ENCRYPT_METHOD SHA512@' \
       -e 's@/var/spool/mail@/var/mail@' etc/login.defs
```

Remarque

Si vous compilez Shadow avec le support pour Cracklib, lancez ce qui suit :

```
sed -i 's@DICTPATH.*@DICTPATH\t/lib/cracklib/pw_dict@' \
    etc/login.defs
```

Préparez la compilation de Shadow :

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

Compilez le paquet :

```
make
```

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

Installez le paquet :

```
make install
```

Déplacez un programme mal placé au bon endroit :

```
mv -v /usr/bin/passwd /bin
```

6.26.2. Configuration de Shadow

Ce paquet contient des outils pour ajouter, modifier, supprimer des utilisateurs et des groupes, initialiser et changer leur mots de passe, et bien d'autres tâches administratives. Pour une explication complète de ce que signifie *password shadowing*, jetez un œil dans le fichier `doc/HOWTO` à l'intérieur du répertoire source. Il reste une chose à garder à l'esprit si vous décidez d'utiliser le support de Shadow : les programmes qui ont besoin de vérifier les mots de passe (gestionnaires d'affichage, programmes FTP, démons pop3 et ainsi de suite) ont besoin d'être *compatibles avec shadow*, c'est-à-dire qu'ils ont besoin d'être capables de fonctionner avec des mots de passe shadow.

Pour activer les mots de passe shadow, lancez la commande suivante :

```
pwconv
```

Pour activer les mots de passe shadow pour les groupes, lancez :

```
grpconv
```

La configuration fournie avec Shadow pour l'outil 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 a 1000. Cela signifie que si vous ne passez pas de paramètres à **useradd**, chaque utilisateur sera membre d'un groupe unique sur le système. Si vous ne désirez pas ce comportement, vous devrez passer le paramètre *`-g`* à **useradd**. Les paramètres par défaut sont stockés dans fichier `/etc/default/useradd`. Il se peut que vous deviez modifier deux paramètres dans ce fichier pour satisfaire vos besoins particuliers.

`/etc/default/useradd` Explication de parametres

`GROUP=1000`

Ce paramètre initialise le début des numéros de groupe utilisés dans le fichier /etc/group. Vous pouvez le modifier avec ce que vous désirez. Remarquez que **useradd** ne réutilisera jamais un UID ou un GID. Si le numéro identifié dans ce paramètre est utilisé, il utilisera le numéro disponible suivant celui-ci. Remarquez aussi que si vous n'avez pas de groupe 1000 sur votre système la première fois que vous utilisez **useradd** sans le paramètre *`-g`*, vous obtiendrez un message sur le terminal qui dit : `useradd: unknown GID 1000`. Vous pouvez passer ce message et le numéro de groupe 1000 sera utilisé.

`CREATE_MAIL_SPOOL=yes`

Il résulte de ce paramètre que **useradd** crée un fichier de boîte mail pour le nouvel utilisateur créé. **useradd** rendra le groupe `mail` propriétaire de ce fichier avec les droits 0660. Si vous préféreriez que **useradd** ne crée pas ces fichiers de boîte mail, lancez la commande suivante :

```
sed -i 's/yes/no/' /etc/default/useradd
```

6.26.3. Configurer le mot de passe de root

Choisissez un mot de passe pour l'utilisateur *root* et configurez-le avec :

```
passwd root
```

6.26.4. Contenu de Shadow

Programmes installés:	chage, chfn, chgpasswd, chpasswd, chsh, expiry, faillog, gpasswd, groupadd, groupdel, groupmems, groupmod, grpck, grpconv, grpunconv, lastlog, login, logoutd, newgrp, newusers, nologin, passwd, pwck, pwconv, pwunconv, sg (lien vers newgrp), su, useradd, userdel, usermod, vigr (lien vers vipw) et vipw
Répertoire installé:	/etc/default

Descriptions courtes

chage Utilisé pour modifier le nombre maximum de jours entre des modifications obligatoires du mot de passe

chfn Utilisé pour modifier le nom complet de l'utilisateur et quelques autres informations

chgpasswd Utilisé pour mettre à jour des mots de passe en mode ligne de commande (batch)

chpasswd Utilisée pour mettre à jour les mots de passe utilisateur en ligne de commande

chsh Utilisé pour modifier le shell de connexion par défaut d'un utilisateur

expiry Vérifie et renforce la politique d'expiration des mots de passe

faillog Est utilisé pour examiner les traces d'échecs de connexions, pour configurer le nombre maximum d'échecs avant qu'un compte ne soit bloqué ou pour réinitialiser le nombre d'échecs

gpasswd Est utilisé pour ajouter et supprimer des membres et des administrateurs aux groupes

groupadd Crée un groupe avec le nom donné

groupdel Supprime le groupe ayant le nom donné

groupmems Permet à un utilisateur d'administrer la liste des membres de son groupe sans avoir besoin des privilèges du super utilisateur

groupmod Est utilisé pour modifier le nom ou le GID du groupe

grpck Vérifie l'intégrité des fichiers `/etc/group` et `/etc/gshadow`

grpconv Crée ou met à jour le fichier shadow à partir du fichier group standard

grpunconv Met à jour `/etc/group` à partir de `/etc/gshadow` puis supprime ce dernier

lastlog Indique les connexions les plus récentes de tous les utilisateurs ou d'un utilisateur donné

login Est utilisé par le système pour permettre aux utilisateurs de se connecter

logoutd Est un démon utilisé pour renforcer les restrictions sur les temps et ports de connexion

newgrp Est utilisé pour modifier le GID courant pendant une session de connexion

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`

6.27. Coreutils-8.21

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

Temps de construction estimé : 1.4 SBU
Espace disque requis : 116 Mo

6.27.1. Installation de Coreutils

POSIX exige que les programmes de Coreutils reconnaissent les limites des caractères correctement même dans des locales multibyte. Le correctif suivant corrige cette rigidité et d'autres bogues liés à l'internationalisation :

```
patch -Np1 -i ../coreutils-8.21-i18n-1.patch
```

Remarque

Autrefois, on a trouvé beaucoup de bogues dans ce correctif. Lorsque vous signalez aux mainteneurs de Coreutils de nouveaux bogues, merci de vérifier d'abord qu'ils sont reproduisibles sans ce correctif.

Maintenant, préparez la compilation de Coreutils :

```
FORCE_UNSAFE_CONFIGURE=1 ./configure \
            --prefix=/usr         \
            --libexecdir=/usr/lib \
            --enable-no-install-program=kill,uptime
```

Voici la signification des options de configuration.

`--enable-no-install-program=kill,uptime`
Le but de ce paramètre est d'empêcher Coreutils d'installer des binaires qui seront installés plus tard par d'autres paquets.

Compilez le paquet :

```
make
```

Passez à « Installez le paquet » si vous n'exécutez pas la suite de test.

Maintenant, la suite de tests peut être lancée. Tout d'abord, lancez les quelques tests qui ont besoin d'être lancés en tant que `root` :

```
make NON_ROOT_USERNAME=nobody SUBDIRS= check-root
```

Nous allons exécuter le reste des tests en tant qu'utilisateur `nobody`. 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 créer un utilisateur `nobody` à part :

```
echo "dummy:x:1000:nobody" >> /etc/group
```

Corrigez des droits afin qu'un utilisateur non-root puisse compiler et exécuter les tests :

```
chown -Rv nobody .
```

Maintenant, lancez les tests. Assurez-vous que PATH dans l'environnement **su** inclut /tools/bin.

```
su nobody -s /bin/bash \
          -c "PATH=$PATH make RUN_EXPENSIVE_TESTS=yes check"
```

Supprimez le groupe temporaire :

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

Installez le paquet :

```
make install
```

Déplacez quelques programmes aux emplacements spécifiés par le FHS :

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

Certains des scripts du paquet LFS-Bootscripts dépendent de **head**, **sleep**, et **nice**. Comme `/usr` pourrait ne pas être disponible dans les premières phases du démarrage, ces binaires ont besoin d'être sur la partition root :

```
mv -v /usr/bin/{head,sleep,nice} /bin
```

6.27.2. Contenu de Coreutils

Programmes installés: [, base64, basename, 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, 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
Répertoire installé: /usr/libexec/coreutils

Descriptions courtes

base64 Encode et décode des données selon la spécification de la base64 (RFC 3548)

basename Supprime tout le chemin et un suffixe donné à partir du nom de fichier donné

cat Concatène des fichiers sur la sortie standard

chcon Modifie le contexte de sécurité d'un fichier

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/ou groupe de certains fichiers et répertoires

chroot Lance une commande avec le répertoire spécifié / comme répertoire racine

cksum Affiche la somme de vérification CRC (Cyclic Redundancy Check) et le nombre d'octets de chaque fichier

comm Compare deux fichiers triés, affichant sur trois colonnes, les lignes uniques et les lignes communes

cp Copie des fichiers

csplit Divise un fichier donné sur plusieurs fichiers indiqués, les séparant par des modèles donnés ou des numéros de lignes. Il affiche le nombre total d'octets pour chaque nouveau fichier

cut Affiche des parties de lignes, sélectionnant ces parties suivant des champs ou positions donnés

date Affiche l'heure actuelle dans le format donné ou initialise la date système

dd Copie un fichier en utilisant la taille et le nombre de blocs donnés tout en réalisant des conversions optionnelles

df Affiche l'espace disque disponible (et utilisé) sur tous les systèmes de fichiers montés, ou seulement sur les systèmes de fichiers contenant les fichiers donnés

dir Liste le contenu de chaque répertoire donné (identique à la commande **ls**)

dircolors Affiche les commandes pour initialiser la variable d'environnement LS_COLOR ce qui permet de changer le schéma de couleurs utilisé par **ls**

dirname Supprime le suffixe qui ne représente pas le répertoire dans un nom de fichier donné

du Affiche le total de l'espace disque utilisé par le répertoire actuel, ou par chacun des répertoires donnés incluant tous les sous-répertoires, ou par chacun des fichiers donnés

echo Affiche les chaînes données

env Lance une commande dans un environnement modifié

expand Convertit les tabulations en espaces

expr Évalue des expressions

factor Affiche les facteurs premiers de tous les entiers spécifiés

false Ne fait rien. Il renvoie toujours un code d'erreur indiquant l'échec

fmt Reformate les paragraphes dans les fichiers donnés

fold Emballe les lignes des fichiers donnés

groups Affiche les groupes auxquels appartient un utilisateur

head Affiche les dix premières lignes (ou le nombre demandé de lignes) pour chaque fichier précisé

hostid Affiche l'identifieur numérique de l'hôte (en hexadécimal)

id Affiche l'identifieur effectif de l'utilisateur courant ou de l'utilisateur précisé, l'identifieur du groupe et les groupes auxquels appartient cet utilisateur

install Copie les fichiers en initialisant leur droits et, si possible, leur propriétaire et groupe

join Joint à partir de deux fichiers les lignes qui ont des champs de jointure identiques

link Crée un lien physique avec le nom de donné vers le fichier indiqué

ln Crée des liens symboliques ou physiques entre des fichiers

logname	Indique le nom de connexion de l'utilisateur actuel
ls	Liste le contenu de chaque répertoire donné
md5sum	Affiche ou vérifie les sommes de vérification MD5 (Message Digest 5)
mkdir	Crée des répertoires avec les noms donnés
mkfifo	Crée des fichiers FIFO (First-In, First-Out, un « tube nommé » dans le vocable d'Unix) avec les noms donnés
mknod	Crée des noeuds périphérique avec les noms donnés. Un noeud périphérique est de type caractère ou bloc, ou encore un FIFO
mktemp	Crée des fichiers temporaires de manière sécurisée, il est utilisé dans des scripts
mv	Déplace ou renomme des fichiers ou répertoires
nice	Lance un programme avec une priorité modifiée
nl	Numérote les lignes de fichiers donnés
nohup	Lance une commande immune aux arrêts brutaux, dont la sortie est redirigée vers le journal de traces
nproc	Affiche le nombre d'unités d'action disponibles pour un processus
od	Affiche les fichiers en octal ou sous d'autres formes
paste	Joint les fichiers donnés en plaçant les lignes correspondantes l'une à côté de l'autre, en les séparant par des caractères de tabulation
pathchk	Vérifie que les noms de fichier sont valides ou portables
pinky	Un client « finger » léger. Il affiche quelques informations sur les utilisateurs indiqués
pr	Fait de la pagination, principalement en colonne, des fichiers pour une impression
printenv	Affiche l'environnement
printf	Affiche les arguments donnés suivant le format demandé, un peu comme la fonction C printf
ptx	Produit un index permuté à partir du contenu des fichiers indiqués, avec chaque mot dans son contexte
pwd	Indique le nom du répertoire courant
readlink	Indique la valeur du lien symbolique
realpath	Affiche le chemin résolu
rm	Supprime des fichiers ou des répertoires
rmdir	Supprime des répertoires s'ils sont vides
runcon	Lance une commande avec le contexte de sécurité spécifié
seq	Affiche une séquence de nombres, à l'intérieur d'un intervalle et avec un incrément spécifié
sha1sum	Affiche ou vérifie des sommes de contrôle 160-bit Secure Hash Algorithm (SHA1)
sha224sum	Affiche ou vérifie des sommes de contrôle 224-bit Secure Hash Algorithm (SHA1)
sha256sum	Affiche ou vérifie des sommes de contrôle 256-bit Secure Hash Algorithm (SHA1)
sha384sum	Affiche ou vérifie des sommes de contrôle 384-bit Secure Hash Algorithm (SHA1)
sha512sum	Affiche ou vérifie des sommes de contrôle 512-bit Secure Hash Algorithm (SHA1)

shred Efface les fichiers indiqués en écrivant dessus des modèles aléatoires pour rendre la récupération des données très difficile

shuf Mélange des lignes de texte

sleep Fait une pause d'un certain temps

sort Trie les lignes des fichiers donnés

split Divise les fichiers donnés en plusieurs pièces, par taille ou par nombre de lignes

stat Affiche le statut du fichier ou du système de fichiers

stdbuf Lance des commandes avec des opérations de mise en tampon modifiées pour ses streamings 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'horodotage d'un fichier, initialise les dates/heures d'accès et de modification des fichiers indiqués à l'heure actuelle. Les fichiers inexistants sont créés avec une longueur nulle

tr Convertit, compresse et supprime les caractères lus depuis l'entrée standard

true Ne fait rien mais avec succès. Il quitte avec un code de sortie indiquant une réussite

truncate Réduit ou augmente un fichier selon la taille spécifiée

tsort Réalise un tri topologique. Il écrit une liste totalement ordonnée suivant un fichier donné partiellement ordonné

tty Indique le nom du fichier du terminal connecté à l'entrée standard

uname Affiche des informations système

unexpand Convertit les espaces en tabulations

uniq Ne conserve qu'une seule ligne parmi plusieurs lignes successives identiques

unlink Supprime le fichier donné

users Indique les noms des utilisateurs actuellement connectés

vdir Est identique à **ls -l**

wc Indique le nombre de lignes, mots et octets de chaque fichier indiqué ainsi que le total de lignes lorsque plus d'un fichier est donné

who Indique qui est connecté

whoami Indique le nom de l'utilisateur associé avec l'identifieur utilisateur effectif

yes Affiche indéfiniment « y » ou la chaîne précisée jusqu'à ce que le processus soit tué

`libstdbuf.so` Bibliothèque utilisée par **stdbuf**

6.28. Iana-Etc-2.30

Le paquet Iana-Etc fournit des données pour les services et protocoles réseau.

Temps de construction estimé : moins de 0.1 SBU
Espace disque requis : 2.2 Mo

6.28.1. Installation de Iana-Etc

La commande suivante convertit les données brutes fournies par l'IANA dans les bons formats pour les fichiers de données `/etc/protocols` et `/etc/services` :

```
make
```

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

Installez le paquet :

```
make install
```

6.28.2. 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 leur numéros de port et types de protocoles affectés

6.29. M4-1.4.16

Le paquet M4 contient un processeur de macros.

Temps de construction estimé : 0.4 SBU
Espace disque requis : 26.6 Mo

6.29.1. Installation de M4

Corrigez une incompatibilité entre ce paquet et Glibc-2.17

```
sed -i -e '/gets is a/d' lib/stdio.in.h
```

Préparez la compilation de M4 :

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

Compilez le paquet :

```
make
```

Pour tester les résultats, corrigez d'abord un programme de test puis lancez les programmes de tests :

```
sed -i -e '41s/ENOENT/& || errno == EINVAL/' tests/test-readlink.h
make check
```

Installez le paquet :

```
make install
```

6.29.2. Contenu de M4

Programme installé: m4

Descriptions courtes

m4 Copie les fichiers donnés pendant l'expansion des 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, lancer des commandes Unix, faire des opérations arithmétiques, manipuler du texte de nombreuses façon, connaît la récursion et ainsi de suite. Le programme **m4** peut être utilisé soit comme interface d'un compilateur soit comme processeur de macros dans son espace.

6.30. Bison-2.7

Le paquet Bison contient un générateur d'analyseurs.

Temps de construction estimé : 1.3 SBU
Espace disque requis : 34 Mo

6.30.1. Installation de Bison

Préparez la compilation de Bison :

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

Le système configure provoque le fait que Bison est compilé sans support pour l'internationalisation des messages d'erreur si un programme **bison** n'est pas déjà dans $PATH. L'ajout suivant va corriger cela :

```
echo '#define YYENABLE_NLS 1' >> lib/config.h
```

Compilez le paquet :

```
make
```

Pour tester les résultats (environ 0.5 SBU), lancez :

```
make check
```

Installez le paquet :

```
make install
```

6.30.2. 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.a` La bibliothèque Yacc contenant des implémentations, compatible Yacc, des fonctions `yyerror` et `main` ; cette bibliothèque n'est généralement pas très utile mais POSIX la réclame

6.31. Grep-2.14

Le paquet Grep contient des programmes de recherche à l'intérieur de fichiers.

Temps de construction estimé : 0.4 SBU
Espace disque requis : 30 Mo

6.31.1. Installation de Grep

Préparez la compilation de Grep :

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

Compilez le paquet :

```
make
```

Pour tester les résultats, lancez :

```
make check
```

Installez le paquet :

```
make install
```

6.31.2. Contenu de Grep

Programmes installés: egrep, fgrep et grep

Descriptions courtes

egrep Affiche les lignes correspondant à une expression rationnelle étendue
fgrep Affiche des lignes correspondant à une liste de chaînes fixes
grep Affiche des lignes correspondant à une expression rationnelle basique

6.32. Readline-6.2

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 estimé : 0.1 SBU
Espace disque requis : 17.2 Mo

6.32.1. Installation de Readline

Réinstaller 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
```

Appliquez un correctif pour corriger un bogue connu corrigé en amont :

```
patch -Np1 -i ../readline-6.2-fixes-1.patch
```

Préparez la compilation de Readline :

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

Compilez le paquet :

```
make SHLIB_LIBS=-lncurses
```

Voici la signification de l'option de make :

`SHLIB_LIBS=-lncurses`
Cette option force Readline à se lier à la bibliothèque `libncurses` (en réalité, `libncursesw`).

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

Installez le paquet :

```
make install
```

Maintenant, déplacez les bibliothèques statiques à un emplacement plus appropriées :

```
mv -v /lib/lib{readline,history}.a /usr/lib
```

Ensuite, supprimez les fichiers `.so` dans `/lib` et liez les à nouveau vers `/usr/lib` :

```
rm -v /lib/lib{readline,history}.so
ln -sfv ../../lib/libreadline.so.6 /usr/lib/libreadline.so
ln -sfv ../../lib/libhistory.so.6 /usr/lib/libhistory.so
```

Si désiré, installez la documentation :

```
mkdir   -v       /usr/share/doc/readline-6.2
install -v -m644 doc/*.{ps,pdf,html,dvi} \
                 /usr/share/doc/readline-6.2
```

6.32.2. Contenu de Readline

Bibliothèques installées: libhistory.{a,so} et libreadline.{a,so}
Répertoires installés: /usr/include/readline, /usr/share/readline, /usr/share/doc/readline-6.2

Descriptions courtes

`libhistory` Fournit une interface utilisateur cohérente pour rappeler des lignes dans l'historique

`libreadline` Aide à une cohérence dans l'interface utilisateur pour des programmes discrets qui ont besoin d'une interface en ligne de commande

6.33. Bash-4.2

Le paquet Bash contient le shell Bourne-Again.

Temps de construction estimé : 1.7 SBU
Espace disque requis : 45 Mo

6.33.1. Installation de Bash

Tout d'abord, appliquez le correctif suivant pour corriger divers bogues traités en amont :

```
patch -Np1 -i ../bash-4.2-fixes-11.patch
```

Préparez la compilation de Bash :

```
./configure --prefix=/usr                      \
            --bindir=/bin                      \
            --htmldir=/usr/share/doc/bash-4.2 \
            --without-bash-malloc              \
            --with-installed-readline
```

Voici la signification de l'option de configure :

`--htmldir`
Cette option désigne le répertoire dans lequel la documentation au format HTML sera installée.

`--with-installed-readline`
Ce commutateur indique à Bash d'utiliser la bibliothèque `readline` sur le système plutôt que d'utiliser sa propre version de readline.

Compilez le paquet :

```
make
```

Sautez à « Installation du paquet » si vous n'exécutez pas la suite de test.

Pour préparer les tests, assurez-vous que l'utilisateur `nobody` peut écrire dans l'arborescence des sources :

```
chown -Rv nobody .
```

Maintenant, lancez les tests en tant qu'utilisateur `nobody` :

```
su nobody -s /bin/bash -c "PATH=$PATH make tests"
```

Installez le paquet :

```
make install
```

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

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

Remarque

Les paramètres utilisés font que **bash** lance un shell de connexion interactif et désactive le hachage, de façon à ce que les nouveaux programme soient découverts au fur et à mesure de leur disponibilité.

6.33.2. Contenu de Bash

Programmes installés: bash, bashbug et sh (lien vers bash)
Répertoire installé: /usr/share/doc/bash-4.2

Descriptions courtes

bash Un interpréteur de commandes largement utilisé ; il réalise un grand nombre d'expansions et de substitutions sur une ligne de commande donnée avant de l'exécuter, rendant cet interpréteur très puissant

bashbug Un script shell pour aider l'utilisateur à composer et à envoyer des courriers électroniques contenant des rapports de bogues spécialement formatés concernant **bash**

sh Un lien symbolique vers le programme **bash** ; à son appel en tant que **sh**, **bash** essaie de copier le comportement initial des versions historiques de **sh** aussi fidèlement que possible, tout en se conformant aussi au standard POSIX

6.34. Libtool-2.4.2

Le paquet Libtool contient le script de support de bibliothèques génériques GNU. Il emballe la complexité d'utilisation de bibliothèques partagées dans une interface cohérente et portable.

Temps de construction estimé : 3.0 SBU
Espace disque requis : 37 Mo

6.34.1. Installation de Libtool

Préparez la compilation de Libtool :

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

Compilez le paquet :

```
make
```

Pour tester les résultats (environ 3.0 SBU), lancez :

```
make check
```

Installez le paquet :

```
make install
```

6.34.2. Contenu de Libtool

Programmes installés: libtool et libtoolize
Bibliothèques installées: libltdl.{a,so}
Répertoires installés: /usr/include/libltdl, /usr/share/libtool

Descriptions courtes

libtool Fournit des services de support de construction généralisée de bibliothèques
libtoolize Fournit une façon standard d'ajouter le support de **libtool** dans un paquet
`libltdl` Cache les nombreuses difficultés avec dlopen sur les bibliothèques

6.35. GDBM-1.10

Le paquet GDBM contient le *GNU Database Manager* (GNU gestionnaire de bases de données). C'est une base de données de formats de fichiers de disque qui conserve la clé/paires de données (data-pairs) dans un seul fichier. La donnée finale de l'enregistrement conservée est indexée par une clé unique, qui peut être récupérée en moins de temps que si elle était conservée dans un fichier texte.

Temps de construction estimé : 0.1 SBU
Espace disque requis : 8.5 Mo

6.35.1. Installation de GDBM

Préparez la compilation de GDBM :

```
./configure --prefix=/usr --enable-libgdbm-compat
```

Voici la signification de l'option de configuration :

`--enable-libgdbm-compat`
Ce paquet permet à la bibliothèque de compatibilité libgdbm de se construire car 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
```

6.35.2. Contenu de GDBM

Programme installé: testgdbm
Bibliothèques installées: libgdbm.{so,a} and libgdbm_compat.{so,a}

Descriptions courtes

testgdbm Règle et modifie une base de données GDBM
`libgdbm` Contient des fonctions pour manipuler une base de données hachée

6.36. Inetutils-1.9.1

Le paquet Inetutils contient des programmes réseau basiques.

Temps de construction estimé : 0.4 SBU
Espace disque requis : 27 Mo

6.36.1. Installation de Inetutils

Corrigez une incompatibilité entre ce paquet et Glibc-2.17

```
sed -i -e '/gets is a/d' lib/stdio.in.h
```

Préparez la compilation d'Inetutils :

```
./configure --prefix=/usr  \
    --libexecdir=/usr/sbin \
    --localstatedir=/var   \
    --disable-ifconfig     \
    --disable-logger       \
    --disable-syslogd      \
    --disable-whois        \
    --disable-servers
```

Voici la signification des options de configure :

`--disable-ifconfig`
Cette option empêche Inetutils d'installer le programme **ifconfig** qui peut être utilisé pour configurer les interfaces réseau. LFS utilise **ip** de IPRoute2 pour accomplir cette tâche.

`--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 meilleure version plus tard

`--disable-syslogd`
Cette option empêche l'installation du démon de traces système par Inetutils car il est installé avec le paquet Sysklogd.

`--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-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 sont non sécurisés et ne sont pas considérés sains sur des réseaux de confiance. Plus d'informations sont disponibles sur *http://lfs.traduc.org/view/blfs-svn/basicnet/inetutils.html*. 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 certains programmes pour qu'ils soient disponibles si `/usr` n'est pas accessible :

```
mv -v /usr/bin/{hostname,ping,ping6} /bin
mv -v /usr/bin/traceroute /sbin
```

6.36.2. Contenu de Inetutils

Programmes installés: ftp, hostname, ping, ping6, rcp, rexec, rlogin, rsh, talk, telnet, tftp, traceroute

Descriptions courtes

ftp	Est un programme de transfert de fichier
hostname	Affiche ou règle le nom de l'hôte
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
rcp	Fait une copie de fichiers distants
rexec	Exécute des commandes sur une machine distante
rlogin	Permet une connexion à distance
rsh	Exécute un shell distant
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

6.37. Perl-5.16.2

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

Temps de construction estimé : 7.5 SBU
Espace disque requis : 247 Mo

6.37.1. Installation de Perl

Tout d'abord, créer un fichier `/etc/hosts` basique pour être référencé dans un des fichiers de configuration de Perl en tant que suite de tests optionnelle :

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

Cette version de Perl compile maintenant le module Compress::Raw::Zlib. Par défaut Perl utilisera une copie interne du code source Zlib pour la compilation. Lancez la commande suivante afin que Perl utilise la bibliothèque Zlib installée sur le système :

```
sed -i -e "s|BUILD_ZLIB\s*= True|BUILD_ZLIB = False|"           \
       -e "s|INCLUDE\s*= ./zlib-src|INCLUDE    = /usr/include|" \
       -e "s|LIB\s*= ./zlib-src|LIB        = /usr/lib|"        \
    cpan/Compress-Raw-Zlib/config.in
```

Si vous voulez avoir un contrôle total sur la façon dont Perl est configuré, vous pouvez supprimer les options « -des » de la commande suivante et contrôler à la main la façon dont ce paquet est construit. Alternativement, utilisez exactement la commande ci-dessous pour utiliser les paramètres par défaut que détecte Perl automatiquement :

```
sh Configure -des -Dprefix=/usr                 \
                  -Dvendorprefix=/usr           \
                  -Dman1dir=/usr/share/man/man1 \
                  -Dman3dir=/usr/share/man/man3 \
                  -Dpager="/usr/bin/less -isR"  \
                  -Duseshrplib
```

Voici la signification de l'option de configure :

`-Dvendorprefix=/usr`
Ceci s'assure que **perl** sait comment dire aux paquets où ils devraient installer leurs modules Perl.

`-Dpager="/usr/bin/less -isR"`
Ceci corrige une erreur dans la façon dont **perldoc** fait appel au programme **less**.

`-Dman1dir=/usr/share/man/man1 -Dman3dir=/usr/share/man/man3`
Comme Groff n'est pas installé, Configure pense que nous ne voulons pas les pages de manuel de Perl. Ces paramètres changent cette décision.

`-Duseshrplib`
Construit une bibliothèque partagée dont certains modules perl ont besoin.

Compilez le paquet :

```
make
```

Pour tester les résultats (approximativement 2.5 SBU), lancez :

```
make -k test
```

Installez le paquet :

```
make install
```

6.37.2. Contenu de Perl

Programmes installés:	a2p, c2ph, config_data, corelist, cpan, cpan2dist, cpanp, cpanp-run-perl, enc2xs, find2perl, h2ph, h2xs, instmodsh, json_pp, libnetcfg, perl, perl5.16.2 (lien vers perl), perlbug, perldoc, perlivp, perlthanks (lien vers perlbug), piconv, pl2pm, pod2html, pod2latex, pod2man, pod2text, pod2usage, podchecker, podselect, prove, psed (lien vers s2p), pstruct (lien vers c2ph), ptar, ptardiff, ptargrep, s2p, shasum, splain, xsubpp et zipdetails
Bibliothèques installées:	Plusieurs centaines qui ne peuvent pas être toutes listées ici
Répertoire installé:	/usr/lib/perl5

Descriptions courtes

a2p	Traduit awk en perl
c2ph	Affiche les structures C comme si elles étaient générées à partir de **cc -g -S**
config_data	Interroge ou modifie la configuration des modules Perl
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
cpan2dist	Le créateur de distribution CPANPLUS
cpanp	Le lanceur CPANPLUS
cpanp-run-perl	Script Perl qui est utilisé pour activer la mise en rouge du tampon de sortie après chaque écriture dans des processus démarrés
enc2xs	Construit une extension Perl pour le module Encode, soit à partir de *Unicode Character Mappings* soit à partir de *Tcl Encoding Files*
find2perl	Traduit les commandes **find** en Perl
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 même 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
perl5.16.2	Un lien vers **perl**

perlbug	Utilisé pour générer des rapports de bogues sur Perl ou les modules l'accompagnant et pour les envoyer par courrier électronique
perldoc	Affiche une partie de la documentation au format pod, embarquée dans le répertoire d'installation de Perl ou dans un script Perl
perlivp	La procédure de vérification d'installation de Perl (*Perl Installation Verification Procedure*). 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
pod2latex	Convertit des fichiers à partir du format pod vers le format LaTeX
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.
psed	Une version Perl de l'éditeur en flux **sed**
pstruct	Affiche les structures C générées à partir de **cc -g -S** stabs
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
s2p	Traduit les scripts **sed** en perl
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

6.38. Autoconf-2.69

Le paquet Autoconf contient des programmes produisant des scripts shell qui configurent automatiquement le code source.

Temps de construction estimé : 4.5 SBU
Espace disque requis : 17.1 Mo

6.38.1. Installation de Autoconf

Préparez la compilation d'Autoconf :

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

Compilez le paquet :

```
make
```

Pour tester les résultats, lancez :

```
make check
```

Ceci prend du temps, pratiquement 4.7 SBUs. En plus, six tests sont ignorés car ils utilisent Automake. Pour effectuer tous les tests, vous pouvez retester Autoconf après que Automake a été installé.

Installez le paquet :

```
make install
```

6.38.2. Contenu de Autoconf

Programmes installés: autoconf, autoheader, autom4te, autoreconf, autoscan, autoupdate et ifnames
Répertoire installé: /usr/share/autoconf

Descriptions courtes

autoconf Produit des scripts shell configurant automatiquement des paquets de code source, permettant ainsi de les adapter à tous les types de systèmes Unix. Les scripts de configuration qu'il produit sont indépendants. Les exécuter ne nécessite pas le programme **autoconf**.

autoheader Un outil pour créer des fichiers modèle d'instructions C *#define* que configure utilise.

autom4te Un emballage pour le processeur de macro M4.

autoreconf Exécute automatiquement **autoconf**, **autoheader**, **aclocal**, **automake**, **gettextize**, et **libtoolize** dans le bon ordre pour gagner du temps lorsque des modifications ont eu lieu sur les fichiers modèles d'**autoconf** et d'**automake**

autoscan Aide à la création de fichiers `configure.in` pour un paquet logiciel. Il examine les fichiers source d'un répertoire et crée un fichier `configure.scan` servant de fichier `configure.in` préliminaire pour le paquet

autoupdate Modifie un fichier `configure.in` qui appelle toujours les macros **autoconf** par leurs anciens noms pour qu'il utilise les noms de macros actuels.

ifnames Sert à écrire les fichiers `configure.in` pour un paquet logiciel. Il affiche les identifieurs que le paquet utilise dans des conditions du préprocesseur C. Si un paquet a déjà été initialisé pour avoir une certaine portabilité, ce programme aide à déterminer ce que **configure** doit vérifier. Il peut aussi remplir les blancs dans un fichier `configure.in` généré par **autoscan**

6.39. Automake-1.13.1

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

Temps de construction estimé : moins de 0.1 SBU (34.1 SBU avec les tests)
Espace disque requis : 100 Mo

6.39.1. Installation de Automake

Préparez la compilation d'Automake :

```
./configure --prefix=/usr --docdir=/usr/share/doc/automake-1.13.1
```

Compilez le paquet :

```
make
```

Remarque

Les tests mettent beaucoup de temps, plus de 30 SBUs. Lancer les tests n'est pas recommandé.

Pour tester les résultats, lancez :

```
make check
```

Installez le paquet :

```
make install
```

6.39.2. Contenu de Automake

Programmes installés: acinstall, aclocal, aclocal-1.13, automake, automake-1.13, compile, config.guess, config.sub, depcomp, elisp-comp, install-sh, mdate-sh, missing, mkinstalldirs, py-compile, symlink-tree et ylwrap
Répertoires installés: /usr/share/aclocal-1.13, /usr/share/automake-1.13, /usr/share/doc/automake-1.13.1

Descriptions courtes

acinstall Un script qui installe des fichiers M4, style aclocal

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

aclocal-1.13 Un lien vers **aclocal**

automake Un outil pour générer automatiquement des fichiers `Makefile.in` à partir de fichiers `Makefile.am`. Pour créer tous les fichiers `Makefile.in` d'un paquet, lancez ce programme dans le répertoire de haut niveau. En parcourant le fichier `configure.in`, il trouve automatiquement chaque fichier `Makefile.am` approprié et génère le fichier `Makefile.in`

automake-1.13 Un lien vers **automake**

compile Un emballage pour les compilateurs

config.guess	Un script qui tente de deviner un triplet canonique pour la construction donnée, l'hôte ou l'architecture de la cible
config.sub	Un script contenant une sous-routine de validation de configuration
depcomp	Un script pour compiler un programme de façon à ce que les informations de dépendances soient générées en plus de la sortie désirée
elisp-comp	Compile le code Lisp d'Emacs
install-sh	Un script qui installe un programme, un script ou un fichier de données
mdate-sh	Un script qui affiche la date de modification d'un fichier ou répertoire
missing	Un script agissant comme remplaçant pour les programmes GNU manquants lors d'une installation
mkinstalldirs	Un script qui crée un ensemble de répertoires
py-compile	Compile un programme Python
symlink-tree	Un script créant un ensemble de liens à partir d'un ensemble de répertoires
ylwrap	Un emballage pour **lex** et **yacc**

6.40. Diffutils-3.2

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

Temps de construction estimé : 0.5 SBU
Espace disque requis : 25 Mo

6.40.1. Installation de Diffutils

Corrigez une incompatibilité entre ce paquet et Glibc-2.17

```
sed -i -e '/gets is a/d' lib/stdio.in.h
```

Préparez la compilation de Diffutils :

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

Compilez le paquet :

```
make
```

Pour tester les résultats, lancez :

```
make check
```

Installez ce paquet :

```
make install
```

6.40.2. 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

6.41. Gawk-4.0.2

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

Temps de construction estimé : 0.2 SBU
Espace disque requis : 30 Mo

6.41.1. Installation de Gawk

Préparez la compilation de Gawk :

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

Compilez le paquet :

```
make
```

Pour tester les résultats, lancez :

```
make check
```

Installez le paquet :

```
make install
```

Si désiré, installez la documentation :

```
mkdir -v /usr/share/doc/gawk-4.0.2
cp    -v doc/{awkforai.txt,*.{eps,pdf,jpg}} /usr/share/doc/gawk-4.0.2
```

6.41.2. Contenu de Gawk

Programmes installés: awk (lien vers gawk), dgawk, gawk, gawk-4.0.2, grcat, igawk, pgawk, pgawk-4.0.2, et pwcat
Répertoires installés: /usr/lib/awk, /usr/share/awk

Descriptions courtes

awk Un lien vers **gawk**
dgawk Un débogueur awk
gawk Un programme de manipulation de fichiers texte. C'est l'implémentation GNU d'**awk**
gawk-4.0.2 Un lien vers **gawk**
grcat Sauvegarde la base de données des groupes, ie `/etc/group`
igawk Donne à **gawk** la capacité d'inclure des fichiers
pgawk La version de profilage de **gawk**
pgawk-4.0.2 Lien vers **pgawk**
pwcat Affiche la base de données de mots de passe `/etc/passwd`

6.42. Findutils-4.4.2

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 estimé : 0.4 SBU
Espace disque requis : 29 Mo

6.42.1. Installation de Findutils

Préparez la compilation de Findutils :

```
./configure --prefix=/usr                     \
            --libexecdir=/usr/lib/findutils \
            --localstatedir=/var/lib/locate
```

Voici la signification de l'option de configure :

`--localstatedir`
Cette option modifie l'emplacement de la base de données **locate** pour qu'elle soit dans `/var/lib/locate`, pour être compatible avec FHS.

Compilez le paquet :

```
make
```

Pour tester les résultats, lancez :

```
make check
```

Installez le paquet :

```
make install
```

Certains scripts du paquet LFS-Bootscripts dépendent de **find**. Comme `/usr` peut ne pas être disponible lors des premières étapes du démarrage, ce programme doit être sur la partition racine. Le script **updatedb** doit aussi être modifié pour corriger un chemin explicite :

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

6.42.2. Contenu de Findutils

Programmes installés: bigram, code, find, frcode, locate, oldfind, updatedb et xargs
Répertoire installé: /usr/lib/findutils

Descriptions courtes

bigram Était auparavant utilisé pour créer les bases de données **locate**

code Était auparavant utilisé pour créer les bases de données **locate** ; c'est l'ancêtre de **frcode**.

find Cherche dans les hiérarchies de répertoires donnés les fichiers correspondant à un critère spécifié

frcode Est appelé par **updatedb** pour compacter la liste des noms de fichiers. Il utilise front-compression, réduisant la taille de la base de données d'un facteur de quatre à cinq

locate Recherche à travers la base de données des noms de fichiers et renvoie les noms contenant une certaine chaîne ou correspondant à un certain modèle

oldfind Ancienne version de find, qui utilise un algorithme différent

updatedb Met à jour la base de données **locate** ; Il parcourt le système de fichiers entier (en incluant les autres systèmes de fichiers actuellement montés, sauf si le contraire est spécifié) et place tous les noms de fichiers qu'ils trouvent dans la base de données

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

6.43. Flex-2.5.37

Le paquet Flex contient un outil de génération de programmes reconnaissant des modèles de texte.

Temps de construction estimé : 0.4 SBU
Espace disque requis : 39 Mo

6.43.1. Installation de Flex

Tout d'abord, corrigez certains tests de régression :

```
patch -Np1 -i ../flex-2.5.37-bison-2.6.1-1.patch
```

Préparez la compilation de Flex :

```
./configure --prefix=/usr              \
            --docdir=/usr/share/doc/flex-2.5.37
```

Compilez le paquet :

```
make
```

Pour tester les résultats (environ 0.5 SBU), lancez :

```
make check
```

Installez le paquet :

```
make install
```

Quelques paquets s'attendent à trouver la bibliothèque `lex` dans `/usr/lib`. Créez un lien symbolique pour en tenir compte :

```
ln -sv libfl.a /usr/lib/libl.a
```

Quelques programmes ne connaissent pas encore **flex** et essaient de lancer son prédécesseur, **lex**. Pour ces programmes, créez un script d'emballage nommé `lex` appelant `flex` en mode d'émulation **lex** :

```
cat > /usr/bin/lex << "EOF"
#!/bin/sh
# Begin /usr/bin/lex

exec /usr/bin/flex -l "$@"

# End /usr/bin/lex
EOF
chmod -v 755 /usr/bin/lex
```

6.43.2. Contenu de Flex

Programmes installés: flex, flex++ (lien vers flex) et lex
Bibliothèques installées: libfl.a et libfl_pic.a
Répertoires installés: /usr/share/doc/flex-2.5.37

Descriptions courtes

flex Un outil pour générer des programmes reconnaissant des modèles dans un texte ; cela permet une grande diversité pour spécifier les règles de recherche de modèle, éradiquant ainsi le besoin de développer un programme spécialisé

flex++ Une extension de flex, est utilisée pour générer du code et des classes C++. C'est un lien symbolique vers **flex**

lex Un script qui exécute **flex** en mode d'émulation **lex**

`libfl.a` La bibliothèque `flex`

6.44. Gettext-0.18.2

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

Temps de construction estimé : 2.3 SBU
Espace disque requis : 180 Mo

6.44.1. Installation de Gettext

Préparez la compilation de Gettext :

```
./configure --prefix=/usr \
        --docdir=/usr/share/doc/gettext-0.18.2
```

Compilez le paquet :

```
make
```

Pour tester les résultats, lancez :

```
make check
```

Installez le paquet :

```
make install
```

6.44.2. Contenu de Gettext

Programmes installés: autopoint, config.charset, config.rpath, envsubst, gettext, gettext.sh, gettextize, hostname, msgattrib, msgcat, msgcmp, msgcomm, msgconv, msgen, msgexec, msgfilter, msgfmt, msggrep, msginit, msgmerge, msgunfmt, msguniq, ngettext, recode-sr-latin et xgettext
Bibliothèques installées: libasprintf.{a,so}, libgettextlib.so, libgettextpo.{a,so}, libgettextsrc.so et preloadable_libintl.so
Répertoires installés: /usr/lib/gettext, /usr/share/doc/gettext-0.18.2, /usr/share/gettext

Descriptions courtes

autopoint Copie les fichiers d'infrastructure standard gettext en un paquet source

config.charset Affiche une table des caractères dépendante du système.

config.rpath Affiche un ensemble de variables dépendant du système, décrivant comment initialiser le chemin de recherche à l'exécution des bibliothèques partagées dans un exécutable

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
hostname	Affiche un nom d'hôte réseau sous plusieurs formats
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`
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 fait 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
`preloadable_libintl`	Une bibliothèque faite pour etre utilisee par LD_PRELOAD et qui aide `libintl` a archiver des messages non traduits.

6.45. Groff-1.22.1

Le paquet Groff contient des programmes de formatage de texte.

Temps de construction estimé : 0.5 SBU
Espace disque requis : 83 Mo

6.45.1. Installation de Groff

Groff s'attend à ce que la variable d'environnement *PAGE=letter* soit adéquate. *PAGE=A4* pourrait aller mieux ailleurs. Si la taille du papier par défaut est configurée lors de la compilation, elle peut être réécrite plus tard en écrivant « A4 » ou « letter » dans le fichier `/etc/papersize`.

Corrigez d'abord un bogue dans un Makefile qui empêche la bonne création d'un lien symbolique d'un PDF :

```
sed -i -e '163 s/$(DESTDIR)//' contrib/mom/Makefile.sub
```

Maintenant, préparez la compilation de Groff :

```
PAGE=<taille_papier> ./configure --prefix=/usr
```

Compilez le paquet :

```
make
```

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

Installez le paquet :

```
mkdir -p /usr/share/doc/groff-1.22/pdf
make install
```

Quelques programmes de documentation, comme **xman**, ne fonctionnent pas correctement sans les liens symboliques suivants :

```
ln -sv eqn /usr/bin/geqn
ln -sv tbl /usr/bin/gtbl
```

6.45.2. Contenu de Groff

Programmes installés: addftinfo, afmtodit, chem, eqn, eqn2graph, gdiffmk, geqn (lien vers eqn), grap2graph, grn, grodvi, groff, groffer, grog, grolbp, grolj4, grops, grotty, gtbl (lien vers tbl), hpftodit, indxbib, lkbib, lookbib, mmroff, neqn, nroff, 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, /usr/share/doc/groff-1.22.1, /usr/share/groff

Descriptions courtes

addftinfo Lit un fichier de polices troff et ajoute quelques informations métriques supplémentaires sur la police qui est utilisée par le système **groff**

afmtodit Crée un fichier de police à utiliser avec **groff** et **grops**

chem Préprocesseur Groff pour produire des diagrammes de structure chimique

eqn Compile les descriptions d'équations imbriquées dans les fichiers d'entrée de troff pour obtenir des commandes comprises par **troff**

eqn2graph Convertit une équation EQN troff en une image améliorée

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

geqn Un lien vers **eqn**

grap2graph Convertit diagramme grap en image bitmap exploitable

grn Un préprocesseur **groff** pour les fichiers gremlin

grodvi Un pilote pour **groff** qui produit un format dvi TeX

groff Une interface au système de formatage de document groff. Normalement, il lance le programme **troff** et un post-processeur approprié au périphérique sélectionné

groffer Affiche des fichiers groff et des pages man sur des terminaux X et tty

grog Lit des fichiers et devine les options `-e`, `-man`, `-me`, `-mm`, `-ms`, `-p`, `-s`, et `-t` de **groff** requises pour l'impression des fichiers. Il indique la commande **groff** incluant ces options

grolbp Pilote **groff** pour les imprimantes Canon CAPSL (imprimantes laser de la série LBP-4 et LBP-8

grolj4 Un pilote pour **groff** produisant une sortie au format PCL5, intéressant les imprimantes HP Laserjet 4

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

gtbl Un lien vers **tbl**

hpftodit Crée un fichier de polices à utiliser avec **groff** -**Tlj4** à partir d'un fichier métrique de police HP

indxbib Crée un index inversé d'un fichier spécifié, index utilisé par les bases de données bibliographiques avec **refer**, **lookbib** et **lkbib**

lkbib Recherche dans les bases de données bibliographiques des références contenant certaines clés et indique toute référence trouvée

lookbib Affiche une invite sur la sortie des erreurs (sauf si l'entrée standard n'est pas un terminal), lit à partir de l'entrée standard une ligne contenant un ensemble de mots clés, recherche dans les bases de données bibliographiques dans un fichier spécifié les références contenant ces mots clés, affiche toute référence trouvée sur la sortie standard et répère ce processus jusqu'à la fin de l'entrée

mmroff Un pré-processeur pour **groff**

neqn Formate les équations pour une sortie ASCII (*American Standard Code for Information Interchange*)

nroff Un script qui émule la commande **nroff** en utilisant **groff**

pdfroff Crée des documents pdf en utilisant groff

pfbtops Traduit une police Postscript au format `.pfb`

pic Compile les descriptions d'images embarquées à l'intérieur de fichiers d'entrées troff ou TeX en des commandes comprises par TeX ou **troff**

pic2graph	Convertit un diagramme PIC en une image améliorée
post-grohtml	Traduit la sortie de GNU **troff** en HTML
preconv	Convertit l'encodage de fichiers en entrée vers quelque chose que comprend GNU **troff**
pre-grohtml	Traduit la sortie de GNU **troff** en HTML
refer	Copie le contenu d'un fichier sur la sortie standard, sauf pour les lignes entre les symboles *.[* et *.]* interprétées comme des citations, et les lignes entre *.R1* et *.R2* interprétées comme des commandes sur la façon de gérer les citations
roff2dvi	Transforme des fichiers roff au format DVI
roff2html	Transforme des fichiers roff au format HTML
roff2pdf	Transforme des fichiers roff au format PDF
roff2ps	Transforme des fichiers roff au format ps
roff2text	Transforme des fichiers roff en fichiers textes
roff2x	Transforme des fichiers roff dans d'autres formats
soelim	Lit des fichiers et remplace les lignes de la forme *file*
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

6.46. Xz-5.0.4

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 estimé : 0.3 SBU
Espace disque requis : 18 Mo

6.46.1. Installation de Xz

Préparez la compilation de Xz :

```
./configure --prefix=/usr --libdir=/lib --docdir=/usr/share/doc/xz-5.0.4
```

Compilez le paquet :

```
make
```

Pour tester les résultats, exécutez :

```
make check
```

Installez le paquet :

```
make pkgconfigdir=/usr/lib/pkgconfig install
```

6.46.2. Contenu de Xz

Programmes installés: lzcat (link to xz), lzcmp (link to xzdiff), lzdiff (link to 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 (link to 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, xzmore
Bibliothèques installées: liblzma.{a,so}
Répertoires installés: /usr/include/lzma et /usr/share/doc/xz-5.0.4

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 **egrep** 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

6.47. GRUB-2.00

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

Temps de construction estimé : 0.7 SBU
Espace disque requis : 112 Mo

6.47.1. Installation de GRUB

Corrigez une incompatibilité entre ce paquet et Glibc-2.17

```
sed -i -e '/gets is a/d' grub-core/gnulib/stdio.in.h
```

Préparez la compilation de GRUB :

```
./configure --prefix=/usr          \
            --sysconfdir=/etc      \
            --disable-grub-emu-usb \
            --disable-efiemu       \
            --disable-werror
```

L'option --disable-werror permet à la construction de se terminer malgré les avertissements apparus avec la dernière version de flex. Les autres paramètres --disable minimisent ce qui est construit en en désactivant des fonctionnalités et des programmes de test pas vraiment nécessaires pour LFS.

Compilez le paquet :

```
make
```

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

Installez le paquet :

```
make install
```

L'utilisation de GRUB pour rendre un système LFS amorçable sera traitée au Section 8.4, « Utiliser GRUB pour paramétrer le processus de démarrage ».

6.47.2. Contenu de GRUB

Programmes installés: grub-bios-setup, grub-editenv, grub-fstest, grub-install, grub-kbdcomp, 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-script-check, grub-set-default, grub-sparc64-setup
Répertoires installés: /usr/lib/grub, /etc/grub.d, /usr/share/grub, /boot/grub

Descriptions courtes

grub-bios-setup Est un programme d'aide pour grub-install
grub-editenv Un outil pour éditer l'ensemble d'environnement
grub-fstest Outil de débogage du pilote d'un système de fichiers

grub-install	Installe GRUB sur votre lecteur
grub-kbdcomp	Script qui convertit un plan xkb dans un plan reconnu par GRUB
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-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

6.48. Less-451

Le paquet Less contient un visualisateur de fichiers texte.

Temps de construction estimé : moins de 0.1 SBU
Espace disque requis : 3.8 Mo

6.48.1. Installation de Less

Préparez la compilation de Less :

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

Voici la signification de l'option de configure :

`--sysconfdir=/etc`
Cette option indique aux programmes créés par le paquet de chercher leurs fichiers de configuration dans `/etc`.

Compilez le paquet :

```
make
```

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

Installez le paquet :

```
make install
```

6.48.2. Contenu de Less

Programmes installés: less, lessecho et lesskey

Descriptions courtes

less Un visualisateur de fichiers. Il affiche le contenu du fichier donné, vous permettant d'aller vers le haut et vers le bas, de chercher des chaînes et de sauter vers des repères

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

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

6.49. Gzip-1.5

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

Temps de construction estimé : 0.2 SBU
Espace disque requis : 19.7 Mo

6.49.1. Installation de Gzip

Préparez la compilation de Gzip :

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

Compilez le paquet :

```
make
```

Pour tester les résultats, lancez :

```
make check
```

Installez le paquet :

```
make install
```

Déplacez des programmes qui n'ont pas besoin d'être sur le système de fichiers racine :

```
mv -v /bin/{gzexe,uncompress,zcmp,zdiff,zegrep} /usr/bin
mv -v /bin/{zfgrep,zforce,zgrep,zless,zmore,znew} /usr/bin
```

6.49.2. Contenu de Gzip

Programmes installés: gunzip, gzexe, gzip, uncompress, 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	Convertit les fichiers formatés avec **compress** au format **gzip**— de `.Z` vers `.gz`

6.50. IPRoute2-3.8.0

Le paquet IPRoute2 contient des programmes pour le réseau, basique ou avancé, basé sur IPV4.

Temps de construction estimé : 0.1 SBU
Espace disque requis : 7.3 Mo

6.50.1. Installation de IPRoute2

Le binaire **arpd** inclu dans ce paquet dépend de Berkeley DB. **arpd** n'étant pas un besoin vraiment courant sur un système Linux de base, supprimez la dépendance de Berkeley DB en appliquant la commande **sed** 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://lfs.traduc.org/view/blfs-svn/server/databases.html#db*.

```
sed -i '/^TARGETS/s@arpd@@g' misc/Makefile
sed -i /ARPD/d Makefile
sed -i 's/arpd.8//' man/man8/Makefile
```

Supprimez un drapeau du compilateur qui fait échouer la compilation :

```
sed -i 's/-Werror//' Makefile
```

Compilez le paquet :

```
make DESTDIR=
```

Voici la signification de l'option de make :

DESTDIR=
Ceci assure que les binaires IPRoute2 vont s'installer dans le bon répertoire. Par défaut, *DESTDIR* initialise un `/usr`.

Ce paquet est fourni avec une suite de tests, mais à cause de sa nature, il n'est pas possible d'exécuter ces tests de manière fiable à partir de l'environnement chroot. Si vous souhaitez lancer ces tests après avoir démarré dans votre nouveau système LFS, assurez-vous de sélectionner le support pour `/proc/config.gz` CONFIG_IKCONFIG_PROC ("General setup" -> "Enable access to .config through /proc/config.gz") dans votre noyau, puis lancez 'make alltests' depuis le sous-répertoire `testsuite/`.

Installez le paquet :

```
make DESTDIR=              \
     MANDIR=/usr/share/man \
     DOCDIR=/usr/share/doc/iproute2-3.8.0 install
```

6.50.2. 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, /lib/tc, /usr/share/doc/iproute2-3.8.0, /usr/lib/tc

Descriptions courtes

bridge Configure des ponts réseaux

ctstat Outil donnant le statut de la connexion

genl

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 lisibles

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

6.51. Kbd-1.15.5

Le paquet Kbd contient les fichiers de tables de caractères, les polices de la console et des outils pour le clavier.

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

6.51.1. Installation de Kbd

Le comportement des touches Effacement et Supprimer n'est pas logique dans les tables de correspondance du clavier du paquet Kbd. Le correctif suivant répare ce problème pour les tables de correspondance du clavier de i386 :

```
patch -Np1 -i ../kbd-1.15.5-backspace-1.patch
```

Après la correction, la touche Effacement génère le caractère de code 127, et la touche Supprimer génère une séquence d'échappement bien connue.

Corrigez un bogue qui fait que certaines tables du clavier ne sont pas chargées correctement :

```
sed -i -e '326 s/if/while/' src/loadkeys.analyze.l
```

Supprimez le programme **resizecons** redondant (il exige la 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 's/\(RESIZECONS_PROGS=\)yes/\1no/g' configure
sed -i 's/resizecons.8 //' man/man8/Makefile.in
```

Préparez la compilation de Kbd :

```
./configure --prefix=/usr \
    --datadir=/lib/kbd --disable-vlock
```

Voici la signification des options de configuration :

`--datadir=/lib/kbd`
Cette option place les données de type de clavier dans un répertoire qui sera toujours sur la partition racine au lieu du `/usr/share/kbd` par défaut.

`--disable-vlock`
Cette option empêche la construction de l'utilitaire vlock, il requiert la bibliothèque PAM qui n'est pas disponible dans l'environnement chroot.

Compilez le paquet :

```
make
```

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

Installez le paquet :

```
make install
```

Remarque

Pour certaines langues (comme le biélorusse), le paquet Kbd ne fournit pas une table de correspondance utile, puisque le contenu de la table assume 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.

Certains des scripts du paquet LFS-Bootscripts dépendent de **kbd_mode**, **loadkeys**, **openvt**, et de **setfont**. Comme `/usr` peut ne pas être disponible lors des premières étapes du démarrage, ces binaires doivent être sur la partition racine :

```
mv -v /usr/bin/{kbd_mode,loadkeys,openvt,setfont} /bin
```

Si désiré, installez la documentation :

```
mkdir -v /usr/share/doc/kbd-1.15.5
cp -R -v doc/* \
         /usr/share/doc/kbd-1.15.5
```

6.51.2. 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, showconsolefont, showkey, unicode_start et unicode_stop

Répertoire installé: /lib/kbd

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 Un lien vers **psfxtable**

psfgettable Un lien vers **psfxtable**

psfstriptable Un lien vers **psfxtable**

psfxtable	Gére 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 des touches meta du clavier
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

6.52. Kmod-12

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

Temps de construction estimé : 0.1 SBU
Espace disque requis : 30 Mo

6.52.1. Installation de Kmod

Préparez la compilation de Kmod :

```
./configure --prefix=/usr       \
            --bindir=/bin       \
            --libdir=/lib       \
            --sysconfdir=/etc   \
            --disable-manpages  \
            --with-xz           \
            --with-zlib
```

Voici la signification des options de configure :

`--with-*`
Ces options permettent à Kmod de gérer des modules noyau compressés.

`--disable-manpages`
Cette option empêche les pages de manuel de se contruire, vu qu'elles s'appuient sur libxslt, qui n'est pas disponible dans l'environnement chroot.

Compilez le paquet :

```
make
```

Pour tester les résultats, lancez :

```
make check
```

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 pkgconfigdir=/usr/lib/pkgconfig install

for target in depmod insmod modinfo modprobe rmmod; do
  ln -sv ../bin/kmod /sbin/$target
done

ln -sv kmod /bin/lsmod
```

6.52.2. Contenu de Kmod

Programmes installés: depmod (lien vers kmod), insmod (lien vers kmod), kmod, kmod-nolib, lsmod (lien vers kmod), modinfo (liien vers kmod), modprobe (lien vers kmod), et rmmod (liien vers kmod)
Bibliothèques installées: /lib/kmod.so

Descriptions courtes

depmod Crée un fichier de dépendances basé sur les symboles qu'il trouve dans l'ensemble de modules existant ; ce fichier de dépendance est utilisé par **modprobe** pour charger automatiquement les modules requis

insmod Installe un module chargeable dans le noyau en cours d'exécution

kmod Charge et décharge des modules du noyau

`libkmod` Cette bibliothèque est utilisée par d'autres programmes pour charger et décharger des modules noyau

lsmod Liste les modules actuellement chargés

modinfo Examine un fichier objet associé à un module noyau et affiche des informations qu'il peut en tirer

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

rmmod Décharge des modules du noyau en cours d'exécution

6.53. Libpipeline-1.2.2

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

Temps de construction estimé : 0.2 SBU
Espace disque requis : 7.4 Mo

6.53.1. Installation de Libpipeline

Préparez la compilation de Libpipeline :

```
PKG_CONFIG_PATH=/tools/lib/pkgconfig ./configure --prefix=/usr
```

Voici la signification des options de configure :

PKG_CONFIG_PATH
Utilise pkg-config pour obtenir l'emplacement des métadonnées de la bibliothèque de test construite dans Section 5.13, « Check-0.9.9 ».

Compilez le paquet :

```
make
```

Pour tester les résultats, lancez :

```
make check
```

Installez le paquet :

```
make install
```

6.53.2. Contenu de Libpipeline

Bibliothèques installées: libpipeline.so

Descriptions courtes

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

6.54. Make-3.82

Le paquet Make contient un programme pour compiler des paquets.

Temps de construction estimé : 0.4 SBU
Espace disque requis : 11.3 Mo

6.54.1. Installation de Make

Tout d'abord, appliquez des correctifs issus de l'amont :

```
patch -Np1 -i ../make-3.82-upstream_fixes-3.patch
```

Préparez la compilation de Make :

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

Compilez le paquet :

```
make
```

Pour tester les résultats, lancez :

```
make check
```

Installez le paquet :

```
make install
```

6.54.2. 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

6.55. Man-DB-2.6.3

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

Temps de construction estimé : 0.5 SBU
Espace disque requis : 27 Mo

6.55.1. Installation de Man-DB

Préparez la compilation de man-DB :

```
./configure --prefix=/usr                         \
            --libexecdir=/usr/lib                 \
            --docdir=/usr/share/doc/man-db-2.6.3 \
            --sysconfdir=/etc                     \
            --disable-setuid                      \
            --with-browser=/usr/bin/lynx          \
            --with-vgrind=/usr/bin/vgrind         \
            --with-grap=/usr/bin/grap
```

Voici la signification des options de configuration :

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

`--with-...`
Ces trois paramètres sont utilisés pour initialiser quelques programmes par défaut. **lynx** est un navigateur Web en mode console (voir BLFS pour les instructions d'installation), **vgrind** convertit du code source de programme en entrée Groff et **grap** est utile pour la composition de texte de graphes dans les documents Groff. Les programmes **vgrind** et **grap** ne sont normalement pas nécessaires pour la visualisation des pages de manuel. Ils ne font pas partie de LFS ou de BLFS mais vous devriez être capable de les installer vous-même après avoir fini LFS si vous souhaitez faire cela.

Compilez le paquet :

```
make
```

Pour tester les résultats, lancez :

```
make check
```

Installez le paquet :

```
make install
```

6.55.2. Pages de manuel non anglaises dans LFS

Le tableau suivant montre l'encodage présumé avec lequel Man-DB encodera de les pages de manuel installçes dans `/usr/share/man/<ll>`. . En outre, Man-DB détermine correctement si les pages de manuel installées dans ce répertoire seront encodées en UTF-8.

Tableau 6.1. 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
Finois (fi)	ISO-8859-1	Letton (lv)	ISO-8859-13
Français (fr)	ISO-8859-1	Macédonien (mk)	ISO-8859-5
Irlandais (ga)	ISO-8859-1	Polonais (pl)	ISO-8859-2
Galicien (gl)	ISO-8859-1	Roumain (ro)	ISO-8859-2
Indonésien (id)	ISO-8859-1	Russe (ru)	KOI8-R
Islandais (is)	ISO-8859-1	Slovaque (sk)	ISO-8859-2
Italien (it)	ISO-8859-1	Slovénien (sl)	ISO-8859-2
Norvégien Bokmal (nb)	ISO-8859-1	Latin serbe (sr@latin)	ISO-8859-2
Holandais (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
Belarusse (be)	CP1251	Chinois, Singapour (zh_SG)	GBK
Bulgare (bg)	CP1251	Chinois traditionnel, Hong Kong (zh_HK)	BIG5HKSCS
Tchèque (cs)	ISO-8859-2	Chinois traditionnel (zh_TW)	BIG5
Grec (el)	ISO-8859-7		

Remarque

Les pages de manuel dans des langues non comprises dans la liste ne sont pas supportées.

6.55.3. Contenu de Man-DB

Programmes installés: accessdb, apropos (lien vers whatis), catman, lexgrog, man, mandb, manpath, whatis et zsoelim
Bibliothèques installées: libman.so, libmandb.so
Répertoires installés: /usr/lib/man-db, /usr/share/doc/man-db

Descriptions courtes

accessdb Transforme le contenu de la base de données **whatis** en format lisible par un humain

apropos	Recherche la base de données **whatis** et affiche les descriptions courtes des commandes système qui contiennent une chaîne donnée
catman	Crée ou met à jour les pages de manuel préformatées
lexgrog	Affiche des informations en résumé d'une ligne à propos d'une page de manuel donnée
man	Formate et affiche les pages de manuel demandées
mandb	Crée ou met à jour la base de données **whatis**
manpath	Affiche le contenu de $MANPATH ou (si $MANPATH n'est pas paramétré) d'un chemin de recherche convenable basé sur les paramètres de l'environnement de l'utilisateur
whatis	Recherche la base de données **whatis** et affiche les descriptions courtes des commandes système qui contiennent le mot-clé donné sous la forme d'un mot séparé
zsoelim	Lit des fichiers et remplace les lignes de la forme *fichier .so* par le contenu du *fichier* mentionné
`libman`	Contient le support au moment de l'exécution de **man**
`libmandb`	Contient le support au moment de l'exécution de **man**

6.56. Patch-2.7.1

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

Temps de construction estimé : moins de 0.1 SBU
Espace disque requis : 3.4 Mo

6.56.1. 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
```

6.56.2. 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.

6.57. Sysklogd-1.5

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

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

6.57.1. Installation de Sysklogd

Compilez le paquet :

```
make
```

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

Installez le paquet :

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

6.57.2. 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
```

6.57.3. Contenu de Sysklogd

Programmes installés: klogd et syslogd

Descriptions courtes

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

syslogd Trace les messages que les programmes systèmes donnent. Chaque message tracé 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

6.58. Sysvinit-2.88dsf

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

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

6.58.1. Installation de Sysvinit

Lorsque les niveaux d'exécution changent (par exemple, lors de l'arrêt du système), **init** envoie des signaux de fin aux processus qu'**init** a lui-même lancé et qui ne devraient plus s'exécuter dans le nouveau niveau d'exécution. En faisant ceci, **init** affiche des messages comme « Sending processes the TERM signal » (NdT : Envoi du signal TERM aux processus) ce qui semble impliquer qu'il envoie ce signal à tous les processus en cours d'exécution. Pour éviter cette mauvaise interprétation, modifiez les sources pour que ce message soit remplacé par « Sending processes started by init the TERM signal » (NdT : Envoi du signal TERM aux processus lancés par init) :

```
sed -i 's@Sending processes@& configured via /etc/inittab@g' src/init.c
```

Les versions maintenues des programmes **wall**, **mountpoint** et **utmpdump** ont été installées plus haut par Util-linux. Supprimez l'installation de la version de Sysvinit de ces programmes et de leur page de man :

```
sed -i -e '/utmpdump/d' \
       -e '/mountpoint/d' src/Makefile
```

Compilez le paquet :

```
make -C src
```

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

Installez le paquet :

```
make -C src install
```

6.58.2. Contenu de Sysvinit

Programmes installés: bootlogd, fstab-decode, halt, init, killall5, last, lastb (lien vers last), mesg, pidof (lien vers killall5), poweroff (lien vers halt), reboot (lien vers halt), runlevel, shutdown, sulogin et telinit (lien vers 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

killall5 Envoie un signal à tous les processus sauf les processus de sa propre session, de façon à ne pas tuer le shell ayant lancé le script qui l'a appelé

last Affiche le dernier utilisateur connecté (et déconnecté) en cherchant dans le fichier `/var/log/wtmp`. Il peut aussi afficher les démarrages et arrêts du système ainsi que les changements de niveaux d'exécution

lastb Affiche les tentatives échouées de connexions tracées dans `/var/log/btmp`

mesg Contrôle si les autres utilisateurs peuvent envoyer des messages au terminal de l'utilisateur courant

pidof Indique le PID des programmes précisés

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 `/var/run/utmp`

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

sulogin Permet la connexion de `root`. Il est normalement appelé par **init** lorsque le système passe en mono-utilisateur

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

6.59. Tar-1.26

Le paquet Tar contient un programme d'archivage.

Temps de construction estimé : 2.4 SBU
Espace disque requis : 34 Mo

6.59.1. Installation de Tar

Corrigez une incompatibilité entre ce paquet et Glibc-2.17

```
sed -i -e '/gets is a/d' gnu/stdio.in.h
```

Préparez la compilation de Tar :

```
FORCE_UNSAFE_CONFIGURE=1  \
./configure --prefix=/usr \
            --bindir=/bin \
            --libexecdir=/usr/sbin
```

Voici la signification des options de configure :

`FORCE_UNSAFE_CONFIGURE=1`
Ceci oblige les tests 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 partiellement construit, on peut le faire sans problèmes.

Compilez le paquet :

```
make
```

Pour tester les résultats (environ 1 SBU), lancez :

```
make check
```

Installez le paquet :

```
make install
make -C doc install-html docdir=/usr/share/doc/tar-1.26
```

6.59.2. Contenu de Tar

Programmes installés: rmt et tar
Répertoire installé: /usr/share/doc/tar-1.26

Descriptions courtes

rmt Manipule à distance un lecteur de bandes magnétiques via une connexion de communication interprocessus

tar Crée, extrait des fichiers à partir d'archives et liste le contenu d'archives, connues sous le nom d'archives tar

6.60. Texinfo-5.0

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

Temps de construction estimé : 0.6 SBU
Espace disque requis : 101 Mo

6.60.1. Installation de Texinfo

Préparez la compilation de Texinfo :

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

Compilez le paquet :

```
make
```

Pour tester les résultats, lancez :

```
make check
```

Installez le paquet :

```
make install
```

De manière optionnelle, installez les composants appartenant à une installation TeX :

```
make TEXMF=/usr/share/texmf install-tex
```

Voici la signification du paramètre de make :

TEXMF=/usr/share/texmf
La variable TEXMF du Makefile contient l'emplacement de la racine de votre répertoire TeX si, par exemple, un paquet TeX sera 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 re-créé, les commandes suivantes accompliront cette tâche :

```
cd /usr/share/info
rm -v dir
for f in *
do install-info $f dir 2>/dev/null
done
```

6.60.2. Contenu de Texinfo

Programmes installés: info, infokey, install-info, makeinfo, pdftexi2dvi, pod2texi, texi2any, texi2dvi, texi2pdf et texindex
Répertoire installé: /usr/share/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**.
infokey	Compile un fichier source contenant des personnalisations Info en un format binaire
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
pod2texi	Convertit du format Pod en Texinfo
texi2any	Traduit une documentation en source Texinfo en d'autres formats
pdftexi2dvi	Utilisé pour formater le document Texinfo donné au format PDF (*Portable Document Format*)
texi2dvi	Utilisé pour formater le document Texinfo indiqué en un fichier indépendant des périphériques, pouvant être édité
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

6.61. Udev-197 (Extrait de systemd-197)

Le paquet Udev contient des programmes pour création dynamique de nœuds de périphériques. Le développement d'udev a été synchronisé avec systemd, mais la plupart de systemd est incompatible avec LFS. Nous ne construisons et n'installons ici que les fichiers d'udev nécessaires.

Temps de construction estimé : 0.1 SBU
Espace disque requis : 23 Mo

6.61.1. Installation d'Udev

Remarque

Ce paquet est un peu différent d'autres paquets. Le paquet de départ extrait est `systemd-197.tar.xz` même si l'application qu'on installe s'appelle udev. Après être allé dans le répertoire systemd, suivez les instructions ci-dessous.

L'archive tar udev-lfs contient les fichiers spécifiques à LFS utilisés pour construire Udev. Déballez-la dans le répertoire des sources de Systemd :

```
tar -xvf ../udev-lfs-197-2.tar.bz2
```

Compilez le paquet :

```
make -f udev-lfs-197-2/Makefile.lfs
```

Installez le paquet :

```
make -f udev-lfs-197-2/Makefile.lfs install
```

Attention

Il existe plusieurs endroits dans le code source de systemd où les chemins des répertoires embarqués sont explicits. Par exemple, la version binaire du chemin de la base de données matériel et le nom de fichier utilisé au moment de l'exécution `/etc/udev/hwdb.bin`, ne peuvent pas être modifiés sans changements explicts dans le code source.

Maintenant, initialisez la base de données du matériel :

```
build/udevadm hwdb --update
```

Enfin, définissez les règles udev du réseau permanent. Cette tâche sera expliquée en détails à la Section 7.2.1, « Création de noms stables pour les interfaces réseaux ». Remarquez que les systèmes de fichiers `/sys` et `/proc` doivent être montés dans l'environnement chroot comme expliqué au début de ce chapitre pour que le script suivant fonctionne.

```
bash udev-lfs-197-2/init-net-rules.sh
```

6.61.2. Contenu d'Udev

Programmes installés: accelerometer, ata_id, cdrom_id, collect, mtd_probe, scsi_id, v4l_id, udevadm et udevd
Bibliothèques installées: libudev.so
Répertoires installés: /etc/udev, /lib/udev, /lib/firmware, /usr/share/doc/udev

Descriptions courtes

ata_id	Fournit Udev avec une chaîne unique et des informations supplémentaires (uuid, label) pour un lecteur ATA
cdrom_id	Fournit Udev avec les possibilités d'un lecteur de CD-ROM ou de DVD-ROM
collect	À partir d'un ID de l'uevent actuel et une liste d'IDs (pour tous les uevents cibles), enregistre l'ID actuel et indique si tous les IDs cibles ont été enregistrés
scsi_id	Fournit Udev avec un identifieur unique de SCSI basé sur les données retournées par l'envoi d'une commande SCSI INQUIRY pour le périphérique spécifié
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 d'uevents, teste la la configuration d'Udev et récupère les 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 droits des périphériques et les règles de périphériques, et les règles pour le nommage des périphériques

6.62. Vim-7.3

Le paquet Vim contient un puissant éditeur de texte.

Temps de construction estimé : 1.1 SBU
Espace disque requis : 96 Mo

Alternatives à Vim

Si vous préférez un autre éditeur—comme Emacs, Joe, ou Nano—merci de vous référer à *http://lfs.traduc.org/view/blfs-svn/postlfs/editors.html* pour des instructions d'installation.

6.62.1. 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
```

Maintenant, préparez la compilation de Vim :

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

Voici la signification de l'option de configure :

`--enable-multibyte`
Ce commutateur optionnel mais hautement recommandé inclut le support pour l'édition de fichiers comprenant des codages de caractères multioctets. Ceci est nécessaire dans le cas d'une utilisation d'une locale avec un ensemble de caractères multi-octets. Ce commutateur peut aussi être utile pour avoir la capacité d'éditer des fichiers créés initialement avec des distributions Linux comme Fedora qui utilise UTF-8 comme encodage par défaut.

Compilez le paquet :

```
make
```

Pour tester les résultats, lancez :

```
make test
```

Néanmoins, cette suite de tests affiche à l'écran beaucoup de caractères binaires qui peuvent causer des soucis sur votre terminal. Ceci peut se résoudre en redirigeant la sortie vers un journal de traces. Un test réussi donnera les mots "ALL DONE" lors de la complétion.

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 est installée dans `/usr/share/vim`. Le lien symbolique suivant permet l'accès à la documentation via `/usr/share/doc/vim-7.3`, le rendant cohérent avec l'emplacement de la documentation pour d'autres paquets :

```
ln -sv ../vim/vim73/doc /usr/share/doc/vim-7.3
```

Si un système X Window va être installé sur votre système LFS, il pourrait être nécessaire de recompiler Vim après avoir installé X. Vim fournit alors une jolie version GUI 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://lfs.traduc.org/view/blfs-svn/postlfs/editors.html#postlfs-editors-vim*.

6.62.2. Configuration de Vim

Par défaut, **vim** est lancé en mode compatible vi. Ceci pourrait être nouveau pour les personnes qui ont utilisé d'autres éditeurs dans le passé. Le paramètre « nocompatible » est inclus ci-dessous pour surligner le fait qu'un nouveau comportement est en cours d'utilisation. Il rappelle aussi à ceux qui voudraient le changer en mode « compatible » qu'il devrait être le premier paramètre dans le fichier de configuration. Ceci est nécessaire car il modifie d'autres paramètres et la surcharge doit survenir après ce paramètre. Créez un fichier de configuration **vim** par défaut en lançant ce qui suit :

```
cat > /etc/vimrc << "EOF"
" Begin /etc/vimrc

set nocompatible
set backspace=2
syntax on
if (&term == "iterm") || (&term == "putty")
  set background=dark
endif

" End /etc/vimrc
EOF
```

L'option *`set nocompatible`* change le comportement de **vim** d'une façon plus utile que le comportement compatible vi. Supprimez « no » pour conserver le comportement de l'ancien **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. 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 meilleurs 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'
```

Remarque

Par défaut, Vim installe des fichiers dictionnaire pour l'anglais. Pour installer des fichiers dictionnaires pour votre langue, téléchargez les fichiers `*.spl` et en option, les `*.sug` pour votre langue et votre encodage sur *ftp://ftp.vim.org/pub/vim/runtime/spell/* et enregistrez-les dans `/usr/share/vim/vim73/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.

6.62.3. 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 couhtes

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

6.63. 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 compilé avec les informations de débogage, le débogueur peut vous donner non seulement les adresses mémoire mais aussi les noms des routine.

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 : 1200 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 utilisés 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.

6.64. Supprimer de nouveau les symboles des fichiers objets

Si l'utilisateur initial n'est pas un développeur et ne pense pas faire de débogage sur les logiciels du système, la taille du système peut être diminué d'environ 200 Mo en supprimant les symboles de débogage contenus dans les binaires et dans les bibliothèques. Ceci ne pose pas de problème autre que le fait de ne plus pouvoir les déboguer.

La plupart des personnes qui utilisent la commande mentionnée ci-dessous ne rencontrent aucune difficulté. Néanmoins, il est facile de faire une erreur de saisie et rendre le nouveau système complètement inutilisable, donc avant d'exécuter la commande **strip**, il est recommandé de faire une sauvegarde de l'état actuel.

Avant d'exécuter la suppression de ces symboles, faites particulièrement attention qu'aucun des binaires concernés ne sont en cours d'exécution. Si vous n'êtes pas sûr que l'utilisateur est entré dans chroot avec la commande donnée dans Section 6.4, « Entrer dans l'environnement chroot, » quittez le chroot :

```
logout
```

Puis, retournez-y avec :

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

Maintenant, les binaires et les bibliothèques peuvent être traitées en toute sécurité :

```
/tools/bin/find /{,usr/}{bin,lib,sbin} -type f \
  -exec /tools/bin/strip --strip-debug '{}' ';'
```

Un grand nombre de fichiers seront rapportés comme ayant un format non reconnu. Ces messages d'avertissement indiquent que ces fichiers sont des scripts et non pas des binaires.

6.65. Nettoyer

À partir de maintenant, en rentrant dans l'environnement chroot après l'avoir quitté, utilisez la commande chroot modifiée suivante :

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

La raison en est que les programmes de `/tools` ne sont plus nécessaires. Comme ils ne sont plus utiles, vous pouvez supprimer le répertoire `/tools` si vous le voulez.

Remarque

Effacer aussi de `/tools` les copies temporaires de Tcl, Expect et DejaGnu, qui ont été utilisées pour lancer les tests de l'ensemble des outils. Si vous avez besoin de ces programmes plus tard, vous devrez les recompiler et les ré-installer. Le livre BLFS a les bonnes instructions pour le faire (voir *http://lfs.traduc.org/view/blfs-svn/*).

Si les systèmes de fichiers virtuel du noyau ont été démontés, manuellement ou suite à un redémarrage, assurez-vous que les systèmes de fichiers virtuels du noyau seront montés lorsque vous entrerez à nouveau dans le chroot. On a expliqué cette procédure dans Section 6.2.2, « Monter et peupler /dev » et Section 6.2.3, « Monter les systèmes de fichiers virtuels du noyau ».

Chapitre 7. Initialiser les scripts de démarrage du système

7.1. Introduction

Ce chapitre traite des fichiers de configuration et des scripts de démarrage. D'abord, il présente les fichiers de configuration généraux nécessaires au paramétrage du réseau :

- Section 7.2, « Configuration générale du réseau. »
- Section 7.3, « Personnaliser le fichier /etc/hosts. »

Ensuite, sont traités les enjeux liés au bon paramétrage des périphériques :

- Section 7.4, « Gestion des périphériques et modules d'un système LFS. »
- Section 7.5, « Création de liens symboliques personnalisés vers les périphériques. »

Les sections suivantes détaillent la manière d'installer et de configurer les scripts du système LFS nécessaires pendant le processus de démarrage. La plupart de ces scripts fonctionneront sans modification, mais il leur faut quelques fichiers de configuration supplémentaires car ils gèrent des informations dépendantes du matériel.

Les scripts de démarrage compatibles System-V sont utilisés dans ce livre parce qu'ils sont largement utilisés et relativement simples. Pour des options supplémentaires, une astuce détaillant le paramétrage du démarrage à la mode BSD est disponible sur *http://www.linuxfromscratch.org/hints/downloads/files/bsd-init.txt*. Une recherche sur les listes de diffusion anglophones de LFS des mots « depinit », « upstart », ou « systemd » donnera également des informations supplémentaires.

Si vous utilisez un autre style de scripts de démarrage, passez ces sections.

Vous pouvez trouver une liste des scripts de démarrage dans l'Annexe D :

- Section 7.6, « LFS-Bootscripts-20130123. »
- Section 7.7, « Comment fonctionnent ces scripts de démarrage ?. »
- Section 7.8, « Configurer le nom d'hôte du système. »
- Section 7.9, « Configurer le script setclock. »
- Section 7.10, « Configurer la console Linux. »
- Section 7.11, « Configurer le script sysklogd. »

Enfin, vous trouverez une courte présentation des scripts et des fichiers de configuration utilisés lorsque l'utilisateur se connecte au système :

- Section 7.13, « Fichiers de démarrage du shell Bash. »
- Section 7.14, « Créer le fichier /etc/inputrc. »

7.2. Configuration générale du réseau

Cette section s'applique seulement si une carte réseau doit être configurée.

Si aucune carte réseau ne sera utilisée, il n'est pas nécessaire de créer des fichiers de configuration relatifs aux cartes réseau. Si c'est le cas, supprimez les liens symboliques `network` de tous les répertoires des niveaux d'exécution (`/etc/rc.d/rc*.d`) après avoir installé les scripts de démarrage de la Section 7.6, « LFS-Bootscripts-20130123 ».

7.2.1. Création de noms stables pour les interfaces réseaux

S'il n'y a qu'une interface réseau à configurer sur le système, cette section est facultative, bien que cela ne fera pas de mal de l'appliquer. Dans de nombreux cas (comme un portable avec deux interfaces l'une sans, l'autre avec fil), l'application de la configuration de cette section est nécessaire.

Avec Udev et les pilotes réseau modulaires, la numérotation des interfaces réseau n'est pas constante entre deux redémarrages, car les pilotes sont chargés en parallèle, et du coup, dans un ordre aléatoire. Par exemple, sur un ordinateur ayant deux cartes réseau fabriquées par Intel et Realtek, la carte réseau produite par Intel peut devenir `eth0` et celle de Realtek `eth1`. Dans certains cas, après un redémarrage, les cartes sont renumérotées d'une autre façon. Pour éviter cela, Udev est fourni avec un script et des règles pour affecter des noms stables aux cartes réseau basés sur leur adresse MAC.

Les règles ont été pré-générées dans les instructions de construction d'udev (systemd) au chapitre précédent. Regardez le fichier `/etc/udev/rules.d/70-persistent-net.rules` pour trouver les noms affectés à vos périphériques réseaux :

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

Remarque

Dans certains cas comme lorsque des adresses MAC ont été affectées à la main à une carte réseau, ou dans un environnement virtuel tel que Xen, il se peut que le fichier de règles réseau n'ait pas été généré car les adresses ne sont pas affectées de manière cohérente. Dans ce cas, passez simplement à la section suivante.

Le fichier commence par un bloc de commentaire suivi de deux lignes pour chaque NIC. La première ligne de chaque NIC est une description commentée des IDs matériels (IDs de fabricant PCI et de périphérique, si c'est une carte PCI), puis de ses pilotes entre parenthèses, si le pilote peut être trouvé. Ni l'ID du périphérique ni le pilote ne sont utilisés pour déterminer quel nom donner à une interface ; Ces informations sont là pour référence seulement. La seconde ligne est la règle Udev qui correspond à ce NIC et qui lui affecte au final un nom.

Toutes les règles Udev sont constituées de plusieurs mots, séparés d'une virgule et optionnellement d'un espace. Ces clés de règle ainsi qu'une explication de chacune d'entre elles sont les suivantes :

- `SUBSYSTEM=="net"` - Demande à Udev d'ignorer les périphériques qui ne sont pas des cartes réseau ;
- `ACTION=="add"` - Demande à Udev d'ignorer cette règle pour un uevent qui n'est pas un ajout (les uevents "retrait" et "changement" se produisent aussi mais ils n'ont pas besoin de renommer les interfaces réseau) ;
- `DRIVERS=="?*"` - Ceci existe afin qu'Udev ignore les VLAN ou les sous-interfaces bridge (car les sous-interfaces n'ont pas de pilotes). Ces sous-interfaces sont ignorées car le nom qui pourrait leur être affecté entrerait en conflit avec leur périphériques parents ;
- `ATTR{adresse}` - La valeur de cette clé est l'adresse MAC du NIC ;
- `ATTR{type}=="1"` - Assure que la règle ne correspond qu'à l'interface primaire dans le cas de certains pilotes sans fil, qui créent plusieurs interfaces virtuelles. Les interfaces secondaires sont ignorées pour la même raison que le sont les VLAN et les sous-interfaces bridge : il y aurait en ce cas un conflit de nom ;

- KERNEL=="eth*" - Cette clé a été ajoutée au générateur de règles d'Udev pour gérer les machines ayant plusieurs interfaces réseau, toutes ayant la même adresse MAC (la PS3 en fait partie). Si les interfaces indépendantes ont des noms de base différents, cette clé permettra à Udev de leur parler en aparté. Ce n'est normalement pas nécessaire pour la plupart des utilisateurs de Linux From Scratch, mais ça ne fait pas de mal ;
- NAME - La valeur de cette clé est le nom qu'Udev affectera à l'interface.

La valeur de NAME est la partie importante. Assurez-vous de connaître quel nom a été affecté à chacune de vos cartes réseau avant de continuer, et assurez-vous d'utiliser cette valeur NAME lorsque vous créerez les fichiers de configuration ci-dessous.

7.2.2. Créer les fichiers de configuration des interfaces réseau

Les interfaces activées et désactivées par le script réseau dépendent des fichiers du répertoire /etc/sysconfig/. Ce répertoire devrait contenir un fichier par interface à configurer, tel que ifconfig.xyz, où « xyz » signifie, pour l'administrateur, quelque chose comme le nom du périphérique (par exemple eth0). Dans ce fichier, il y a les attributs de cette interface, tels que son ou ses adresses IP, ses masques de sous-réseau, et ainsi de suite. Il faut que la fin du nom de fichier soit *ifconfig*.

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=192.168.1.1
GATEWAY=192.168.1.2
PREFIX=24
BROADCAST=192.168.1.255
EOF
```

Les valeurs de ces variables doivent être modifiées dans chaque fichier pour correspondre à la bonne configuration.

Si la variable ONBOOT est configurée à « yes », le script réseau configurera le 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 configurée 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.

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**.

7.2.3. Créer le fichier /etc/resolv.conf

Si le système a besoin d'être connecté à Internet, il aura besoin d'un DNS pour résoudre les noms de domaines Internet en adresse IP, et vice-versa. Ceci se fait en plaçant les adresses IP du serveur DNS, disponibles auprès du FAI ou de l'administrateur système, dans `/etc/resolv.conf`. Créez le fichier en lançant ce qui suit :

```
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.

Remarque

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

7.3. Personnaliser le fichier /etc/hosts

Si une carte réseau doit être configurée, choisissez son 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"
# Début de /etc/hosts (version avec carte réseau)

  127.0.0.1 localhost
  <192.168.1.1> <HOSTNAME.example.org> [alias1] [alias2 ...]

# Fin de /etc/hosts (version avec carte réseau)
EOF
```

Les valeurs *<192.168.1.1>* et *<HOSTNAME.example.org>* doivent être remplacées suivant les contraintes/ besoins des utilisateurs (si la machine se voit affectée 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 optionnels.

Si vous n'avez pas de carte réseau à configurer, créez le fichier /etc/hosts en lançant la commande :

```
cat > /etc/hosts << "EOF"
# Début de /etc/hosts (version sans carte réseau)

127.0.0.1 <HOSTNAME.example.org> <NOMDHOTE> localhost

# Fin de /etc/hosts (version sans carte réseau)
EOF
```

7.4. Gestion des périphériques et modules d'un système LFS

Au Chapitre 6, nous avons installé le paquet Udev. 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 utilisent une méthode de création de périphériques statiques avec laquelle un grand nombre de nœuds de périphériques sont créés sous /dev (quelque fois des milliers de nœuds), que le matériel correspondant existe ou pas. Ceci est fait typiquement 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 pour le périphériques détectés par le noyau sont créés. Comme ces nœuds de périphériques seront créés à chaque lancement du système, ils seront stockés dans un système de fichiers devtmpfs (un système de fichiers virtuel qui réside entièrement dans la mémoire du système). Les nœuds de périphériques ne requièrent pas beaucoup d'espace, donc la mémoire utilisée est négligeable.

7.4.1. Historique

En février 2000, un nouveau système de fichiers appelé devfs a été intégré au noyau 2.3.46 et rendu disponible pour la série 2.4 des noyaux stables. Bien qu'il soit présent dans les sources du noyau, cette méthode de création dynamique des périphériques n'a jamais reçu un support inconditionnel des développeurs du noyau.

Le principal problème de l'approche adopté par devfs était la façon dont il gérait la détection, la création et le nommage des périphériques. Ce dernier problème, le nommage des périphériques, était peut-être le plus critique. Il est généralement accepté que s'il est possible de configurer les noms des périphériques, alors la politique de nommage des périphériques revient à l'administrateur du système, et du coup n'est imposée par aucun développeur en particulier.

Le système de fichiers devfs souffre aussi de restrictions particulières inhérentes à sa conception et qui ne peuvent 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 voir un remplacement de l'espace utilisateur pour devfs est devenu beaucoup plus réaliste.

7.4.2. Implémentation d'Udev

7.4.2.1. 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 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 internes avec sysfs sont disponibles pour les processus en espace utilisateur ainsi que pour **udev** pour continuer (et faire même des modifications aux nœuds de périphériques).

7.4.2.2. 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 devtmpfs (par le cœur du pilote) pour le faire. Quand une instance devtmpfs est montée sur /dev, le nœud de périphérique sera créé dès le départ avec un nom, des droits et un propriétaire figés.

Peu de temps après, le noyau enverra un uevent à **udevd**. À partir des règles indiquées dans les fichiers contenus dans les répertoires /etc/udev/rules.d, /lib/udev/rules.d et /run/udev/rules.d, **udevd** créera les liens symboliques supplémentaires vers le nœud de périphérique, ou bien il modifiera ses droits, son propriétaire ou son groupe, ou l'entrée dans la base de données interne d'**udevd** concernant cet objet.

Les règles de ces trois répertoires sont numérotées de la même façon que dans le paquet LFS-Bootscripts et les trois répertoires sont mis à jour ensemble. 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.

7.4.2.3. Les scripts de démarrage d'Udev

Le premier script de démarrage de LFS, /etc/init.d/mountvirtfs, va copier les périphériques de /lib/udev/devices vers /dev. C'est nécessaire car certains périphériques, certains répertoires et certains liens symboliques sont requis avant que les processus de gestion dynamique des périphériques ne soient disponibles au tout début du démarrage d'un système ou car ils sont exigés par **udevd** lui-même. La création de nœuds de périphériques statiques dans /lib/udev/devices offre aussi un contournement facile pour les périphériques qui ne sont pas supportés par l'infrastructure de gestion dynamique des périphériques.

Le script de démarrage /etc/rc.d/init.d/udev démarre **udevd**, récupère tous les périphériques "montés à froid" ayant déjà été créés par le noyau et attend les règles pour se terminer. Le script défait aussi le gestionnaire d'uevent de son paramétrage par défaut /sbin/hotplug . Cela se fait car le noyau n'a plus besoin d'appeler de binaires externes. À la place, **udevd** listera sur une socket netlink les uevents que le noyau détecte.

Le script de démarrage **/etc/rc.d/init.d/udev_retry** prend soin de ratraper les événements pour les sous-systèmes dont les règles peuvent s'appuyer sur des systèmes de fichiers non montés jusqu'à ce que le script **mountfs** ne s'exécute (en particulier, `/usr` et `/var` peut causer cela). Ce script se lance après le script **mountfs**, afin que les règles (si récupérées), réussissent la deuxième fois. Il est configuré à partir du fichier `/etc/sysconfig/udev_retry` ; tous les mots de ce fichier différents de commentaires sont considérés comme des noms de sous-systèmes pour les récupérer lors de la nouvelle tentative. Pour trouver le sous-système d'un périphérique, utilisez **udevadm info --attribute-walk <périphérique>** où <périphérique> est un chemin absolu dans /dev ou /sys tel que /dev/sr0 ou /sys/class/rtc.

7.4.2.4. 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 spécifiques du bus des périphériques supportés par un module. Par exemple, le pilote *snd-fm801* supporte 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 ». Il résultera des règles fournies par défaut qu'**udevd** fera appel à **/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 support pour des systèmes de fichiers et des NLS sur demande.

7.4.2.5. Gestion des périphériques dynamiques ou montables à chaud

Quand vous connectez un périphérique, comme un lecteur MP3 USB, le noyau reconnaît que le périphérique est maintenant connecté et génère un uevent. Cet uevent est alors géré par **udevd** comme décrit ci-dessus.

7.4.3. Problèmes avec le chargement des modules et la création des périphériques

Il existe quelques problèmes connus pour la création automatique des nœuds de périphériques :

7.4.3.1. Un module noyau n'est pas chargé automatiquement

Udev ne chargera un module que s'il a un alias spécifique au bus et que le pilote du bus envoie correctement les alias nécessaires vers `sysfs`. Sinon, il faut organiser le chargement des modules par d'autres moyens. Avec Linux-3.8.1, 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 a le support nécessaire pour 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 supporte le périphérique et peut lui parler directement, mais s'il n'a pas d'alias, c'est un bogue dans le pilote. Chargez le pilote sans l'aide d'Udev et attendez que le problème soit corrigé plus tard.

S'il n'y a pas de fichier `modalias` dans le bon répertoire sous `/sys/bus`, cela signifie que les développeurs du noyau n'ont pas encore ajouté de support modalias à ce type de bus. Avec Linux-3.8.1, c'est le cas pour les bus ISA. Attendez que ce problème soit réparé dans les versions ultérieures du noyau.

Udev n'a pas du tout pour but de charger des pilotes « wrapper » (qui emballent un autre pilote) comme *snd-pcm-oss* et des pilotes non matériels comme *loop*.

7.4.3.2. Un module du noyau n'est pas chargé automatiquement et Udev n'est pas prévu pour le charger

Si le module « wrapper » 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 le wrapper après qu'Udev ait chargé le module emballé. 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.

7.4.3.3. Udev charge un module indésirable

Ne compilez pas le module, ou mettez-le en liste noire dans un fichier `/etc/modprobe.d/blacklist.conf` comme nous l'avons fait avec le module *forte* dans l'exemple ci-dessous :

```
blacklist forte
```

Les modules en liste noire peuvent toujours être chargés manuellement avec la commande explicite **modprobe**.

7.4.3.4. 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**

7.4.3.5. Une règle Udev fonctionne de manière non fiable

Cela peut être une autre manifestation du problème précédent. Sinon, et si votre règle utilise les attributs de `sysfs`, il se peut que ce soit un problème de timing du noyau, sur le point d'être corrigé dans les noyaux ultérieurs. Pour le moment, vous pouvez contourner en créant une règle qui attend l'attribut `sysfs` utilisé et en le mettant dans le fichier `/etc/udev/rules.d/10-wait_for_sysfs.rules` (créez ce fichier s'il n'existe pas). Merci d'informer la liste de développement de LFS si vous faites ainsi et que cela vous aide.

7.4.3.6. Udev ne crée pas de périphérique

Le texte ci-après suppose que le pilote est compilé de manière statique dans le noyau ou qu'il est déjà chargé comme module, et que vous avez déjà vérifié qu'Udev ne crée pas de périphérique mal nommé.

Udev n'a pas besoin d'information pour créer un nœud de périphérique si le pilote du noyau n'envoie pas ses données vers `sysfs`. C'est ce qu'il y a de plus courant avec les pilotes tierce partie à l'extérieur de l'arborescence du noyau. Créez un nœud de périphérique statique dans `/lib/udev/devices` avec les numéros majeurs/mineurs appropriés (voir le fichier `devices.txt` dans la documentation du noyau ou la documentation fournie par le fabricant du pilote tierce partie). Le nœud du périphérique statique sera copié vers `/dev` par le script de démarrage **udev**.

7.4.3.7. 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 nombre ou la sortie de divers utilitaires *_id installés par Udev. Voir la Section 7.5, « Création de liens symboliques personnalisés vers les périphériques » et la Section 7.2, « Configuration générale du réseau » pour des exemples.

7.4.4. 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`)

7.5. Création de liens symboliques personnalisés vers les périphériques

7.5.1. Liens symboliques pour le CD-ROM

Certains logiciels que vous pourriez vouloir installer plus tard (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 de mettre 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 avez.

Il y a des avantages dans 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 et/ou les slots 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 remplaceriez par un périphérique différent avec les mêmes possibilités 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 changera. 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, et 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 de ID_MODEL et de ID_REVISION. Le mode « chemin » utilisera la valeur de 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 -i -e 's/"write_cd_rules"/"write_cd_rules mode"/' \

    /etc/udev/rules.d/83-cdrom-symlinks.rules
```

Remarquez qu'il n'est pas nécessaire de créer les fichiers de règle 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.

7.5.2. Gestion des périphériques dupliqués

Comme expliqué à la Section 7.4, « Gestion des périphériques et modules d'un système LFS », 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 séparé dans Section 7.2, « Configuration générale du réseau », et vous pouvez trouver la configuration des cartes son dans *BLFS*.

Pour chacun des périphériques susceptibles d'avoir ce problème (même si le problème n'apparaît pas dans votre distribution Linux actuelle), trouvez le répertoire correspondant sous `/sys/class` ou `/sys/block`. Pour les périphériques vidéo, cela peut être `/sys/class/video4linux/videoX`. Calculez les attributs qui identifient de façon unique un périphérique (normalement basé sur l'ID du fabricant et du produit et/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"

# Liens symboliques permanent vers la webcam et le 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 il y a des liens symboliques `/dev/tvtuner` et `/dev/webcam` qui pointent toujours vers le bon périphérique.

7.6. LFS-Bootscripts-20130123

Le paquet LFS-Bootscripts contient un ensemble de scripts pour démarrer ou arrêter le système LFS lors de l'amorçage ou de l'arrêt.

Temps de construction estimé : moins de 0.1 SBU
Espace disque requis : 256 Ko

7.6.1. Installation de LFS-Bootscripts

Installez le paquet :

```
make install
```

7.6.2. Contenu de LFS-Bootscripts

Scripts installés: checkfs, cleanfs, console, functions, halt, ifdown, ifup, localnet, modules, mountfs, mountkernfs, network, rc, reboot, sendsignals, setclock, 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 `/var/run/` et `/var/lock/` ; il recrée `/var/run/utmp` et supprime les fichiers `/etc/nologin`, `/fastboot` et `/forcefsck`.

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 courants.

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

mountkernfs Monte les systèmes de fichiers virtuels fournies 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.

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 vers `/etc/udev/rules.d` si nécessaire.

7.7. Comment fonctionnent ces scripts de démarrage ?

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, du coup, il ne peut pas être supposé que parce que cela fonctionne sur une distribution Linux particulière, cela fonctionnera de la même façon dans LFS. LFS a sa propre façon de le faire mais il respecte généralement les standards établis.

SysVinit (qui sera nommé par la suite « init ») fonctionne en utilisant un schéma de niveaux d'exécution. Ils sont au nombre de sept (numérotés de 0 à 6). En fait, il en existe plus mais ils sont réservés à des cas spéciaux et ne sont généralement pas utilisés. Voir `init(8)` pour plus de détails. Chacun d'entre eux correspond à des actions que l'ordinateur est supposé effectuer lorsqu'il démarre. Le niveau d'exécution par défaut est 3. Voici les descriptions sur l'implémentation des différents niveaux d'exécution :

0: arrête l'ordinateur
1: mode mono-utilisateur
2: mode multi-utilisateur sans réseau
3: mode multi-utilisateur avec réseau
4: réservé pour la personnalisation, sinon identique à 3
5: identique à 4, il est habituellement utilisé pour la connexion GUI (comme **xdm** de X ou **kdm** de KDE)
6: redémarre l'ordinateur

7.7.1. 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"
# Début de /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:S016:once:/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

# Fin de /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/rcsysinit.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.

7.7.2. 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**.

Il existe un certain nombre de répertoires sous `/etc/rc.d` qui ressemble à `rc?.d` (où ? est le numéro du niveau d'exécution) et `rcsysinit.d`, tous contenant un certain nombre de liens symboliques. Certains commencent avec un *K*, les autres avec un *S*, et tous ont deux nombres après la lettre initiale. Le K signifie l'arrêt (kill) d'un service et le S son lancement (start). Les nombres déterminent l'ordre dans lequel les scripts sont exécutés, de 00 à 99 — plus ce nombre est petit, plus tôt le script correspondant sera exécuté. Quand **init** bascule sur un autre niveau d'exécution, les services appropriés sont soit lancés soit tués, suivant le niveau d'exécution choisi.

Les vrais scripts sont dans `/etc/rc.d/init.d`. Ils font le vrai boulot 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`*.

Il existe une exception à cette explication. Les liens commençant avec un *S* dans les répertoires `rc0.d` et `rc6.d` ne lanceront aucun service. Ils seront appelés avec l'argument *`stop`* pour arrêter quelque chose. La logique derrière ceci est que, quand un utilisateur va redémarrer ou arrêter le système, rien ne doit être lancé. Le système a seulement besoin d'être arrêté.

Voici les descriptions de ce que font les arguments des scripts :

`start`
: Le service est lancé.

`stop`
: Le service est stoppé.

`restart`
: Le service est stoppé puis relancé.

`reload`
: La configuration du service est mise à jour. Ceci est utilisé après 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 ici sont un exemple d'une façon de faire.

7.8. Configurer le nom d'hôte du système

Une partie de la tâche du script **localnet** est de configurer le nom du système. Ce nom doit être indiqué dans le fichier `/etc/sysconfig/network`.

Créez le fichier `/etc/sysconfig/network` et saisissez le nom du système en lançant :

```
echo "HOSTNAME=<lfs>" > /etc/sysconfig/network
```

<lfs> doit être remplacé par le nom de l'ordinateur. Ne saisissez pas le FQDN (voir Annexe A). Cette information sera saisie dans le fichier `/etc/hosts`.

7.9. Configurer le script setclock

Le script **setclock** lit le temps sur l'horloge matérielle, aussi connu sous le nom d'horloge BIOS or CMOS. Si l'horloge matérielle est configurée en UTC, ce script convertira le temps de l'horloge matérielle en temps local en utilisant le fichier `/etc/localtime` (indiquant au programme **hwclock** le fuseau horaire où se situe l'utilisateur). Il n'existe pas de moyens de détecter si l'horloge matérielle est configurée en UTC, donc elle doit être configurée manuellement.

setclock est lancé via udev quand le noyau détecte la capacité du matériel au démarrage. Il peut aussi être lancé 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 configurée en UTC, découvrez-le en exécutant **`hwclock --localtime --show`**. Ceci affichera l'heure courante suivant l'horloge matérielle. Si l'heure correspond à ce qui vous dit votre montre, alors l'horloge matérielle est configurée sur l'heure locale. Si la sortie de **hwclock** n'est pas l'heure locale, il y a des chances qu'elle soit configurée en UTC. Vérifiez ceci en ajoutant ou en soustrayant le bon nombre d'heures pour votre fuseau horaire à l'heure affichée par **hwclock**. Par exemple, si vous êtes actuellement sur le fuseau horaire MST, aussi connu en tant que GMT -0700, ajoutez sept heures à l'heure locale.

Modifiez la valeur de la variable `UTC` ci-dessous par une valeur *`0`* (zéro) si l'horloge matérielle n'est *pas* configurée en temps 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

# Réglez ceci à toute option que vous pourriez devoir donner à hwclock,
# comme le type de l'horloge matérielle de la machine pour Alphas.
CLOCKPARAMS=

# Fin de /etc/sysconfig/clock
EOF
```

Une bonne astuce expliquant comment gérer l'horloge sur LFS est disponible sur *http://lfs.traduc.org/view/astuces/heure.txt*. Elle explique certains concepts comme les fuseaux horaires, UTC et la variable d'environnement `TZ`.

Remarque

Vous pouvez régler les paramètres CLOCKPARAMS et UTC dans le fichier `/etc/sysconfig/rc.site`.

7.10. Configurer la console Linux

Cette section discute de la configuration du script de démarrage **console**, initialisant la disposition du clavier, la police de la console et le niveau de journalisation du noyau. Si des caractères non ASCII (par exemple, les symboles copyright, livre anglaise et Euro) ne seront pas utilisés et que le clavier est américain, passez cette section. Sans le fichier de configuration, (ou son équivalent 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 du plan de codage 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 le répertoire `/lib/kbd` pour des plans de codage valides et des polices d'écran. Lisez les pages de manuel `loadkeys(1)` et `setfont(8)` pour 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**. 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 du plan de codage à charger, comme « es ». Si cette variable n'est pas réglée, le script de démarrage ne lancera pas le programme **loadkeys** et le plan de codage du noyau par défaut sera utilisé.

KEYMAP_CORRECTIONS
: Cette variable (rarement utilisée) spécifie les arguments du second appel au programme **loadkeys**. C'est utile si le plan de codage stocké n'est pas totalement satisfaisant et que vous devez faire un petit ajustement. Par exemple, pour inclure le signe Euro dans un plan de codage qui ne l'a normalement pas, réglez cette variable à « euro2 ».

FONT
: Cette variable spécifie les arguments du programme **setfont**. En principe, ceci inclut le nom de la police, « -m » et le nom du plan de caractères de l'application à charger. Par exemple, pour charger la police « lat1-16 » avec le plan de caractères 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 le plan de caractères de l'application pour la conversion de codes de touche 8-bits composés dans le plan de codage en UTF-8, et ainsi vous devriez initialiser l'argument du paramètre "-m" à l'encodage des codes de touche composés dans le plan de codage.

UNICODE
: Réglez cette variable à « 1 », « yes » ou « true » afin de mettre la console en mode UTF-8. Ceci est utile pour les locales basées sur UTF-8 et nuisible sinon.

LEGACY_CHARSET
: Pour beaucoup de types de clavier, il n'y a pas de plan de codage pour le stock Unicode dans le paquet Kbd. Le script de démarrage **console** convertira un plan de codage disponible en UTF-8 au vol si cette variable est réglée à l'encodage du plan de codage 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 un plan de codage stocké. L'exemple suivant ajoute le symbole Euro au plan de codage allemand :

```
cat > /etc/sysconfig/console << "EOF"
# Début de /etc/sysconfig/console

KEYMAP="de-latin1"
KEYMAP_CORRECTIONS="euro2"
FONT="lat0-16 -m 8859-15"

# Fin de /etc/sysconfig/console
EOF
```

- Ce qui suit est un exemple avec l'Unicode activé pour le bulgare, où un plan de codage UTF-8 stocké 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-glyph LatArCyrHeb-16 dans l'exemple précédent, les couleurs brillantes ne sont plus disponibles sur la console Linux à moins qu'un framebuffer soit utilisé. Si vous voulez avoir les couleurs brillantes sans framebuffer et que vous pouvez vivre sans caractère n'appartenant pas à votre langue, il est encore possible d'utiliser une police 256-glyph 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 du plan 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
```

- Certains plans de codage ont des touches mortes (c-à-d que les touches qui 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 le plan de codage par défaut). Linux-3.8.1 n'interprète correctement les touches mortes et les règles de composition que quand les caractères source qui seront composés ensembles sont multi-octet. Ce défaut n'affecte pas les plans de clavier pour les langues européennes, car il y a des accents 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 devraient 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).

Remarque

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 limitations mentionnées dans les deux derniers points de la liste ci-dessus ne s'appliquent pas.

7.11. Configurer le script sysklogd

Le script `sysklogd` invoque le programme **syslogd** avec l'option *`-m 0`*. Cette option 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.

7.12. Le fichier rc.site

Le fichier facultatif `/etc/sysconfig/rc.site` contient les paramètres réglés automatiquement pour chaque script de démarrage. 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 ont la préséance.

`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@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}"

# 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

# Speed up boot without waiting for settle in udev_retry
#OMIT_UDEV_RETRY_SETTLE=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
#LOGLEVEL=5

# 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=
```

7.12.1. 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 bidouillages 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 pas 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 sauter, 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. Ceci car 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 délai 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 sauter la vérification du système de fichiers, **fsck**, complètement. Pour cela, soit créez le fichier `/fastboot`, soit redémarrez 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 délai important dans le processus de démarrage. Pour sauter la suppression de ces fichiers, définissez la variable SKIPTMPCLEAN=y.
- Lors de l'extinction, 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 sauté en définissant le paramètre KILLDELAY=0.

7.13. 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-à-d `[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 au support des langues maternelles. 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 ;
- La 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 supportées 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 ». Quelques 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 donnée 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 la locale, la monnaie et le préfixe du pays à composer avant de saisir le 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 locale n'a pas été installée au chapitre 6 ou qu'elle n'est pas prise en charge par l'installation par défaut de Glibc.

```
locale: Cannot set LC_* to default locale: No such file or directory
```

Si cela arrive, vous 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.

Certains paquets en dehors de LFS pourraient aussi ne pas avoir de support pour la locale que vous avez choisi. Un exemple est la bibliothèque X (qui fait partie du système X Window), qui affiche le message d'erreur suivant si la locale n'est pas exactement identique à celle définie par ses fichiers :

```
Warning: locale not supported by Xlib, locale set to C
```

Dans certains cas Xlib s'attend à ce que le plan de caractères soit listé en majuscule avec des tirets canoniques. Par exemple, "ISO-8859-1" plutôt que "iso88591". Il est aussi possible de trouver la spécification adéquate en supprimant la partie charmap de la spécification de la locale. Vous pouvez le vérifier en lançant la commande **locale charmap** dans les deux locales. Par exemple, vous pourriez vouloir remplacer "fr_FR.ISO-8859-15@euro" par "fr_FR@euro" afin que cette locale soit reconnue par Xlib.

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 supportent 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 locales « C » (par défaut) et « en_US » (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 dans cette locale. De même, essayer d'envoyer un mail avec de tels caractères depuis Mutt ou Pine donne l'envoi de messages en version non compatible avec RFC (le codage du mail sortant est indiqué comme « 8-bit inconnu »). Donc, vous ne pouvez utiliser la locale « 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 supportées par beaucoup de programmes. Le travail progresse pour documenter et, si possible, réparer de tels problèmes, voir *http://lfs.traduc.org/view/blfs-svn/introduction/locale-issues.html*.

7.14. Créer le fichier /etc/inputrc

Le fichier `inputrc` gère les fichiers de correspondance du clavier pour les situations spécifiques. Ce fichier est le fichier de démarrage utilisé par Readline — la bibliothèque relative aux entrées — utilisée par Bash et la plupart des autres shells.

La plupart des personnes n'ont pas besoin de fichiers de correspondance spécifiques, donc la commande ci-dessous crée un fichier `/etc/inputrc` global utilisé par tous ceux qui se connectent. Si vous décidez plus tard que vous avez besoin de surcharger les valeurs par défaut utilisateur par utilisateur, vous pouvez créer un fichier `.inputrc` dans le répertoire personnel de l'utilisateur avec les correspondances modifiées.

Pour plus d'informations sur l'édition du fichier `inputrc`, voir **info bash** à la section *Fichier d'initialisation Readline* (ou *Readline Init File*). **info readline** est aussi une bonne source d'informations.

Ci-dessous se trouve un fichier `inputrc` générique avec des commentaires expliquant l'utilité des différentes options. Remarquez que les commentaires ne peuvent pas être sur la même ligne que les commandes. Créez le fichier en utilisant la commande suivante :

```
cat > /etc/inputrc << "EOF"
# Début de /etc/inputrc
# Modifié par Chris Lynn <roryo@roryo.dynup.net>

# Permettre à l'invite de commande d'aller à la ligne
set horizontal-scroll-mode Off

# Activer l'entrée sur 8 bits
set meta-flag On
set input-meta On

# Ne pas supprimer le 8ème bit
set convert-meta Off

# Conserver le 8ème bit à l'affichage
set output-meta On

# none, visible or audible
set bell-style none

# Toutes les indications qui suivent font correspondre la séquence
# d'échappement contenue dans le 1er argument à la fonction
# spécifique de readline
"\eOd": backward-word
"\eOc": forward-word

# Pour la console linux
"\e[1~": beginning-of-line
"\e[4~": end-of-line
"\e[5~": beginning-of-history
"\e[6~": end-of-history
"\e[3~": delete-char
"\e[2~": quoted-insert

# Pour xterm
"\eOH": beginning-of-line
"\eOF": end-of-line

# Pour Konsole
"\e[H": beginning-of-line
"\e[F": end-of-line

# Fin de /etc/inputrc
EOF
```

Chapitre 8. Rendre le système LFS amorçable

8.1. 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.

8.2. 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"
# Début de /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

# Fin de /etc/fstab
EOF
```

Remplacez *<xxx>*, *<yyy>* et *<fff>* par les valeurs appropriées pour votre système, par exemple `hda2`, `hda5` et `ext3`. 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 de l'option de montage « iocharset » afin d'interpréter correctement les caractères non ASCII dans les noms de fichier. La valeur de cette option 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 trouvez sous File systems -> Native Language Support) a été compilée en dur dans le noyau ou en module. 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,iocharset=koi8r,codepage=866
```

Le fragments d'options correspondantes pour les utilisateurs ru_RU.UTF-8 est :

```
noauto,user,quiet,showexec,iocharset=utf8,codepage=866
```

Remarque

Dans ce dernier cas, le noyau émet le message suivant :

```
FAT: utf8 is not a recommended IO charset for FAT filesystems,
    filesystem will be case sensitive!
```

Vous devriez ignorer cette recommendation négative, puisque toutes les autres valeurs de l'option « iocharset » aboutissent à un mauvais affichage des noms de fichier avec les locales 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 supporte 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 supportée.

Remarque : Les partitions basées sur *Logical Volume Management* (LVM) ne peuvent pas utiliser l'option `barrier`.

8.3. Linux-3.8.1

Le paquet Linux contient le noyau Linux.

Temps de construction estimé : 1.0 - 5.0 SBU
Espace disque requis : 540 - 800 Mo

8.3.1. 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.

Configurez le noyau via l'interface par menu. Pour des informations d'ordre général sur la configuration du noyau, consultez *http://lfs.traduc.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 : *http://lfs.traduc.org/view/blfs-svn/longindex.html#kernel-config-index*.

Remarque

Suite à de récents changements dans udev, assurez-vous de sélectionner :

```
Device Drivers  --->
  Generic Driver Options  --->
     Maintain a devtmpfs filesystem to mount at /dev
```

```
make LANG=<valeur_LANG_du_hote> LC_ALL= menuconfig
```

Voici la signification des paramètres 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 de menuconfig soit correctement dessinée sur la console texte de Linux en UTF-8.

Assurez-vous de remplacer *<valeur_LANG_de_l_hote>* par la valeur de la variable `$LANG` de votre hôte. Si elle n'est pas paramétrée, vous pouvez utiliser à la place les valeurs `$LC_ALL` ou `$LC_CTYPE` de l'hôte.

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-3.8.1` 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 7.4, « Gestion des périphériques et modules d'un système LFS » et dans le répertoire `linux-3.8.1/Documentation` de la documentation du noyau. Enfin, `modprobe.conf(5)` pourrait aussi être intéressant.

Installez les modules si la configuration du noyau les utilise :

```
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`.

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 nommenclature 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 -v arch/x86/boot/bzImage /boot/vmlinuz-3.8.1-lfs-7.3
```

`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 -v System.map /boot/System.map-3.8.1
```

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 -v .config /boot/config-3.8.1
```

Installez la documentation du noyau Linux :

```
install -d /usr/share/doc/linux-3.8.1
cp -r Documentation/* /usr/share/doc/linux-3.8.1
```

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.

Si vous conservez l'arborescence des sources du noyau, lancez **chown -R 0:0** sur le répertoire `linux-3.8.1` 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` devraient *toujours* être celles avec lesquelles Glibc a été compilé, à savoir, les en-têtes expurgées de cette archive du noyau Linux. Donc, elles ne devraient *jamais* être remplacés par les en-têtes du noyau ou par d'autres en-têtes expurgées du noyau.

8.3.2. 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"
# Begin /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
```

8.3.3. Contenu de Linux

Fichiers installés: config-3.8.1, vmlinux-3.8.1-lfs-7.3-3.8.1 et System.map-3.8.1
Répertoires installés: /lib/modules, /usr/share/doc/linux-3.8.1

Descriptions courtes

`config-3.8.1` Contient toutes les options de configuration choisies pour le noyau.

`vmlinuz-3.8.1-lfs-7.3` 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 multitâches capable d'exécuter plusieurs programmes quasi simultanément.

`System.map-3.8.1` 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.

8.4. Utiliser GRUB pour paramétrer le processus de démarrage

8.4.1. 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, c-à-d 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
```

8.4.2. 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. Notez que ceci diffère des versions précédentes où les deux numéros commençaient à zéro. Par exemple, les partitions `sda1` et `sdb3` correspondent pour GRUB à *(hd0,1)* et *(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)*.

8.4.3. 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 100 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 séparemment, 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 distincte 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
```

8.4.4. Créer le fichier de configuration

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 3.8.1-lfs-7.3" {
        linux   /boot/vmlinuz-3.8.1-lfs-7.3 root=/dev/sda2 ro
}
EOF
```

Remarque

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 9. Fin

9.1. 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.

Une bonne idée serait de créer un 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 7.3 > /etc/lfs-release
```

C'est aussi une bonne idée de créer un fichier pour afficher 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="7.3"
DISTRIB_CODENAME="<votre nom ici>"
DISTRIB_DESCRIPTION="Linux From Scratch"
EOF
```

Assurez-vous de personnaliser le champ 'DISTRIB_CODENAME' pour que ce système ne soit que le vôtre.

9.2. Enregistrez-vous

Maintenant que vous avez terminé le livre, voulez-vous être enregistré comme utilisateur de LFS ? Allez directement sur *http://www.linuxfromscratch.org/cgi-bin/lfscounter.php* et enregistrez-vous comme utilisateur LFS en entrant votre nom et la première version de LFS que vous ayez utilisée.

Redémarrons sur LFS maintenant.

9.3. Redémarrer le système

Maintenant que tous les logiciels ont été installés, il est temps de redémarrer votre ordinateur. Néanmoins, vous devez savoir certaines choses. Le système que vous avez créé dans ce livre est vraiment minimaliste et a toutes les chances de ne pas avoir les fonctionnalités dont vous aurez besoin pour continuer. En installant quelques paquets supplémentaires à partir du livre BLFS en restant dans l'environnement chroot actuel, vous serez dans une bien meilleure position pour continuer une fois que vous aurez redémarré votre nouvelle installation LFS. Voici quelques suggestions :

- Un navigateur web en mode texte tel que *Lynx* vous permettra de lire facilement le livre BLFS dans un terminal virtuel tout en construisant les paquets dans un autre.
- Le paquet *GPM* vous permettra de réaliser des copier/coller dans vos terminaux virtuels.
- Si vous êtes dans une situation où la configuration IP statique ne correspond pas à vos besoins en terme de réseau, installer des paquets comme *dhcpcd* ou la partie client de *dhcp* peut être utile.
- Installer *sudo* peut être utile pour construire des paquets en tant qu'utilisateur non root et pour installer facilement les paquets qui en résultent dans votre noureau système.
- Si vous voulez accéder à votre nouveau système depuis un environnement graphique confortable, installez *openssh* et son prérequis *openssl*.

- Pour faicliter le rapatriement de fichiers par Internet, installez *wget*.
- Si vous avez un ou plusieurs disques ayant une table de partitions GUID (GPT), *gptfdisk* ou *parted* seront utiles.
- 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/network
  - /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/shm

if [ -h $LFS/dev/shm ]; then
   link=$(readlink $LFS/dev/shm)
   umount -v $LFS/$link
   unset link
else
   umount -v $LFS/dev/shm
fi
umount -v $LFS/dev
umount -v $LFS/proc
umount -v $LFS/sys
```

Démontez le système de fichiers LFS lui-même :

```
umount -v $LFS
```

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
```

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 7.3*.

Quand le redémarrage est terminé, le système LFS est fonctionnel et davantage de logiciels peuvent être installés pour satisfaire vos besoins.

9.4. 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 :

  - Freecode (*http://freecode.com/*)

    Freecode peut vous prévenir (par email) des nouvelles versions des paquets installés sur votre système.

  - *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://lfs.traduc.org/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://www.linuxfromscratch.org/hints/list.html*.

- 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 - Listes de diffusion 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 IV. 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 Semiconducteur 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 Serveur 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

FQDN *Fully Qualified Domain Name* ou Nom de domaine pleinement qualifié

FTP Protocole de transfert de fichiers

GB Giga-octet

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 interprocessus
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
KB Kilo-octet
LED *Light Emitting Diode* ou Diode électroluminescente
LFS Linux From Scratch
LSB Linux Standard Base
MB Mega-octet
MBR *Master Boot Record* ou Secteur d'amorçage
MD5 Message Digest 5
NIC Network Interface Card
NLS Support de la langue maternelle
NNTP *Network News Transport Protocol* ou Protocole de transfert UseNet
NPTL Native POSIX Threading Library
OSS Open Sound System
PCH Pre-Compiled Headers
PCRE Perl Compatible Regular Expression
PID Identificateur de processus
PTY Pseudo terminal
QOS *Quality Of Service* ou Qualité de service
RAM *Random Access Memory* ou Mémoire vive
RPC *Remote Procedure Call* ou Appel de procédure distante
RTC *Real Time Clock* ou Horloge temps réel
SBU Standard Build Unit
SCO The Santa Cruz Operation
SHA1 Secure-Hash Algorithm 1
TLDP The Linux Documentation Project
TFTP *Trivial File Transfer Protocol* ou Protocole simplifié de transfert de fichiers
TLS *Thread-Local Storage* ou Mémoire locale de thread
UID Identificateur utilisateur
umask user file-creation mask
USB Universal Serial Bus

UTC Temps universel coordonné

UUID Identificateur universellement unique

VC Console Virtuelle

VGA Adaptateur graphique vidéo

VT Terminal virtuel

Annexe B. Remerciements

Nous aimerions remercier les personnes et organisations suivantes pour leurs contributions au projet Linux From Scratch.

- *Gerard Beekmans* <gerard@linuxfromscratch.org> – créateur de LFS, leader du projet LFS
- *Matthew Burgess* <matthew@linuxfromscratch.org> – leader du projet LFS, rédacteur technique LFS/éditeur
- *Bruce Dubbs* <bdubbs@linuxfromscratch.org> – gestionnaire des versions de LFS, rédacteur technique LFS/ éditeur
- *Jim Gifford* <jim@linuxfromscratch.org> – Co-Leader du projet CLFS
- *Bryan Kadzban* <bryan@linuxfromscratch.org> – rédacteur technique LFS
- *Randy McMurchy* <randy@linuxfromscratch.org> – Leader du projet BLFS, éditeur LFS
- *DJ Lucas* <dj@linuxfromscratch.org> – éditeur de LFS et de BLFS
- *Ken Moffat* <ken@linuxfromscratch.org> – éditeur LFS et CLFS
- *Ryan Oliver* <ryan@linuxfromscratch.org> – Co-Leader du projet CLFS
- 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
- *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 mirroirs

Mirroirs Nord-Américains

- *Scott Kveton* <scott@osuosl.org> – mirroir lfs.oregonstate.edu
- *William Astle* <lost@l-w.net> – mirroir ca.linuxfromscratch.org
- *Eujon Sellers* <jpolen@rackspace.com> – mirroir lfs.introspeed.com
- *Justin Knierim* <tim@idge.net> – mirroir lfs-matrix.net

Mirroirs Sud-américains

- *Manuel Canales Esparcia* <manuel@linuxfromscratch.org> – mirroir lfsmirror.lfs-es.info
- *Luis Falcon* <Luis Falcon> – mirroir torredehanoi.org

Mirroirs européens

- *Guido Passet* <guido@primerelay.net> – mirroir nl.linuxfromscratch.org
- *Bastiaan Jacques* <baafie@planet.nl> – mirroir lfs.pagefault.net

- *Sven Cranshoff* <sven.cranshoff@lineo.be> – mirroir lfs.lineo.be
- Scarlet Belgium – mirroir lfs.scarlet.be
- *Sebastian Faulborn* <info@aliensoft.org> – mirroir lfs.aliensoft.org
- *Stuart Fox* <stuart@dontuse.ms> – mirroir lfs.dontuse.ms
- *Ralf Uhlemann* <admin@realhost.de> – mirroir lfs.oss-mirror.org
- *Antonin Sprinzl* <Antonin.Sprinzl@tuwien.ac.at> – mirroir at.linuxfromscratch.org
- *Fredrik Danerklint* <fredan-lfs@fredan.org> – mirroir se.linuxfromscratch.org
- *Franck* <franck@linuxpourtous.com> – mirroir lfs.linuxpourtous.com
- *Philippe Baqué* <baque@cict.fr> – mirroir lfs.cict.fr
- *Vitaly Chekasin* <gyouja@pilgrims.ru> – mirroir lfs.pilgrims.ru
- *Benjamin Heil* <kontakt@wankoo.org> – mirroir lfs.wankoo.org

Mirroirs asiatiques

- *Satit Phermsawang* <satit@wbac.ac.th> – mirroir lfs.phayoune.org
- *Shizunet Co.,Ltd.* <info@shizu-net.jp> – mirroir lfs.mirror.shizu-net.jp
- *Init World* <http://www.initworld.com/> – mirroir lfs.initworld.com

Mirroirs australiens

- *Jason Andrade* <jason@dstc.edu.au> – mirroir au.linuxfromscratch.org

Anciens membres de l'équipe du projet

- *Christine Barczak* <theladyskye@linuxfromscratch.org> – éditeur du livre LFS
- Archaic <archaic@linuxfromscratch.org> – rédacteur technique LFS/éditeur, leader du projet HLFS, éditeur de BLFS, mainteneur des projets d'astuces et correctifs
- *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
- 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

- *Dan Nicholson* <dnicholson@linuxfromscratch.org> – éditeur de LFS et BLFS
- *Alexander E. Patrakov* <alexander@linuxfromscratch.org> – rédacteur Technique LFS, éditeur de LFS international, mainteneur du LiveCD LFS
- Simon Perreault
- *Scot Mc Pherson* <scot@linuxfromscratch.org> – mainteneur de LFS NNTP Gateway
- *Greg Schafer* <gschafer@zip.com.au> – rédacteur technique LFS et architecte de la nouvelle méthode de construction activant le 64 bits
- Jesse Tie-Ten-Quee – rédacteur technique LFS
- *James Robertson* <jwrober@linuxfromscratch.org> – mainteneur Bugzilla
- *Tushar Teredesai* <tushar@linuxfromscratch.org> – éditeur du livre BLFS, leader du projet d'astuces et correctifs
- *Jeremy Utley* <jeremy@linuxfromscratch.org> – rédacteur technique LFS, Maintaineur Bugzilla, Mainteneur de LFS-Bootscripts
- *Zack Winkles* <zwinkles@gmail.com> – rédacteur technique LFS

Annexe C. Dépendances

La compilation et l'installation corrects de chaque paquet compilé dans LFS dépend d'un ou plusieurs autres paquets. Certains paquets participent même aux dépendances circulaires, c'est-à-dire que le premier paquet dépend du second qui dépend à son tour du premier. À cause de ces dépendances, l'ordre dans lequel les paquets sont compilés dans LFS est très important. Le but de cette page est de documenter les dépendances de chaque paquet compilé dans LFS.

Pour chaque paquet que nous compilons, nous avons listé trois, parfois quatre types de dépendances. La première concerne les autres paquets qui doivent être disponibles afin de compiler et d'installer le paquet en question. La deuxième concerne les paquets qui, en plus de ceux de la première liste, doivent être disponibles afin de lancer les suites de test. La troisième liste de dépendances contient les paquets qui exigent ce paquet pour être compilés et installés à l'emplacement final avant qu'ils ne soient compilés et installés. Dans la plupart des cas, c'est parce que ces paquets lieront les chemins aux binaires à l'intérieur de leurs scripts. S'ils ne sont pas compilés dans un certain ordre, ceci pourrait aboutir à ce que des chemins vers /tools/bin/[binaire] soient placés à l'intérieur de scripts installés dans le système final. Cela n'est évidemment pas souhaitable.

La dernière indique les dépendances facultatives qui ne sont pas destinées à LFS mais qui pourraient être utiles à l'utilisateur. Ces paquets peuvent avoir eux-mêmes des dépendances supplémentaires obligatoires ou facultatives. Pour ces dépendances, la pratique recommandée consiste à les installer après avoir terminé le livre LFS puis à revenir en arrière pour reconstruire le paquet LFS. Dans certains cas, la réinstallation est traitée dans BLFS.

Autoconf

Dépendances de l'installation :	Bash, Coreutils, Grep, M4, Make, Perl, Sed et Texinfo
Dépendances de la suite de tests :	Automake, Diffutils, Findutils, GCC et Libtool
Doit être installée préalablement :	Automake
Dépendances facultatives:	Emacs

Automake

Dépendances de l'installation :	Autoconf, Bash, Coreutils, Gettext, Grep, M4, Make, Perl, Sed et Texinfo
Dépendances de la suite de tests :	Binutils, Bison, Bzip2, DejaGNU, Diffutils, Expect, Findutils, Flex, GCC, Gettext, Gzip, Libtool et Tar.
Doit être installée préalablement :	Aucune
Dépendances facultatives:	Aucune

Bash

Dépendances de l'installation : Bash, Binutils, Bison, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Make, Ncurses, Patch, Readline, Sed et Texinfo

Dépendances de la suite de tests : Shadow

Doit être installée préalablement : Aucune

Dépendances facultatives: Xorg

Binutils

Dépendances de l'installation : Bash, Binutils, Coreutils, Diffutils, File, Gawk, GCC, Glibc, Grep, Make, Perl, Sed, Texinfo et Zlib

Dépendances de la suite de tests : DejaGNU et Expect

Doit être installée préalablement : Aucune

Dépendances facultatives: Aucune

Bison

Dépendances de l'installation : Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, M4, Make et Sed

Dépendances de la suite de tests : Diffutils et Findutils

Doit être installée préalablement : Flex, Kbd et Tar

Dépendances facultatives: Doxygen (suite de tests)

Bzip2

Dépendances de l'installation : Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Make et Patch

Dépendances de la suite de tests : Aucune

Doit être installée préalablement : Aucune

Dépendances facultatives: Aucune

Coreutils

Dépendances de l'installation : Bash, Binutils, Coreutils, GCC, Gettext, Glibc, GMP, Grep, Make, Patch, Perl, Sed et Texinfo

Dépendances de la suite de tests : Diffutils, E2fsprogs, Findutils, Shadow et Util-linux

Doit être installée préalablement : Bash, Diffutils, Findutils, Man-DB et Udev

Dépendances facultatives: Perl Expect et IO:Tty modules (pour la suites de tests)

DejaGNU

Dépendances de l'installation : Bash, Coreutils, Diffutils, GCC, Grep, Make et Sed

Dépendances de la suite de tests : Pas de suite de tests disponible

Doit être installée préalablement : Aucune

Dépendances facultatives: Aucune

Diffutils

Dépendances de l'installation : Bash, Binutils, Coreutils, Gawk, GCC, Gettext, Glibc, Grep, Make, Sed et Texinfo

Dépendances de la suite de tests : Diffutils, Perl

Doit être installée préalablement : Aucune

Dépendances facultatives: Aucune

Expect

Dépendances de l'installation : Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Grep, Make, Patch, Sed et Tcl

Dépendances de la suite de tests : Aucune

Doit être installée préalablement : Aucune

Dépendances facultatives: Aucune

E2fsprogs

Dépendances de l'installation : Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Gzip, Make, Sed, Texinfo et Util-linux

Dépendances de la suite de tests : procps-ng, Psmisc

Doit être installée préalablement : Aucune

Dépendances facultatives: Aucune

File

Dépendances de l'installation : Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Make, Sed et Zlib

Dépendances de la suite de tests : Aucune

Doit être installée préalablement : Aucune

Dépendances facultatives: Aucune

Findutils

Dépendances de l'installation : Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, Make, Sed et Texinfo

Dépendances de la suite de tests : DejaGNU, Diffutils et Expect

Doit être installée préalablement : Aucune

Dépendances facultatives: Aucune

Flex

Dépendances de l'installation : Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, M4, Make, Patch, Sed et Texinfo

Dépendances de la suite de tests : Bison et Gawk

Doit être installée préalablement : IPRoute2, Kbd et Man-DB

Dépendances facultatives: Aucune

Gawk

Dépendances de l'installation : Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, Make, Patch, Sed and, Texinfo

Dépendances de la suite de tests : Diffutils

Doit être installée préalablement : Aucune

Dépendances facultatives: Aucune

Gcc

Dépendances de l'installation : Bash, Binutils, Coreutils, Diffutils, Findutils, Gawk, GCC, Gettext, Glibc, GMP, Grep, M4, Make, MPC, MPFR, Patch, Perl, Sed, Tar et Texinfo

Dépendances de la suite de tests : DejaGNU et Expect

Doit être installée préalablement : Aucune

Dépendances facultatives: *CLooG-PPL*, *GNAT* et *PPL*

GDBM

Dépendances de l'installation : Bash, Binutils, Coreutils, Diffutils, GCC, Grep, Make et Sed

Dépendances de la suite de tests : Aucune

Doit être installée préalablement : Aucune

Dépendances facultatives: Aucune

Gettext

Dépendances de l'installation : Bash, Binutils, Coreutils, Gawk, GCC, Glibc, Grep, Make, Sed et Texinfo

Dépendances de la suite de tests : Diffutils, Perl et Tcl

Doit être installée préalablement : Automake

Dépendances facultatives: Aucune

Glibc

Dépendances de l'installation : Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Gettext, Grep, Gzip, en-têtes de l'API Linux, Make, Perl, Sed et Texinfo

Dépendances de la suite de tests : File

Doit être installée préalablement : Aucune

Dépendances facultatives: Aucune

GMP

Dépendances de l'installation : Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, M4, Make, Sed et Texinfo

Dépendances de la suite de tests : Aucune

Doit être installée préalablement : MPFR, GCC

Dépendances facultatives: Aucune

Grep

Dépendances de l'installation : Bash, Binutils, Coreutils, Diffutils, GCC, Gettext, Glibc, Grep, Make, Patch, Sed et Texinfo

Dépendances de la suite de tests : Gawk

Doit être installée préalablement : Man-DB

Dépendances facultatives: Pcre, Xorg et CUPS

Groff

Dépendances de l'installation : Bash, Binutils, Bison, Coreutils, Gawk, GCC, Glibc, Grep, Make, Patch, Sed et Texinfo

Dépendances de la suite de tests : Pas de suite de tests disponible

Doit être installée préalablement : Man-DB et Perl

Dépendances facultatives: GPL Ghostscript

GRUB

Dépendances de l'installation : Bash, Binutils, Bison, Coreutils, Diffutils, GCC, Gettext, Glibc, Grep, Make, Ncurses, Sed, Texinfo et Xz

Dépendances de la suite de tests : Aucune

Doit être installée préalablement : Aucune

Dépendances facultatives: Aucune

Gzip

Dépendances de l'installation : Bash, Binutils, Coreutils, GCC, Glibc, Grep, Make, Sed et Texinfo

Dépendances de la suite de tests : Diffutils, less

Doit être installée préalablement : Man-DB

Dépendances facultatives: Aucune

Iana-Etc

Dépendances de l'installation : Coreutils, Gawk et Make

Dépendances de la suite de tests : Pas de suite de tests disponible

Doit être installée préalablement : Perl

Dépendances facultatives: Aucune

Inetutils

Dépendances de l'installation : Bash, Binutils, Coreutils, GCC, Glibc, Grep, Make, Ncurses, Patch, Sed, Texinfo et Zlib

Dépendances de la suite de tests : Pas de suite de tests disponible

Doit être installée préalablement : Tar

Dépendances facultatives: Aucune

IProute2

Dépendances de l'installation : Bash, Bison, Coreutils, Flex, GCC, Glibc, Make et en-têtes de l'API Linux

Dépendances de la suite de tests : Pas de suite de tests disponible

Doit être installée préalablement : Aucune

Dépendances facultatives: Aucune

Kbd

Dépendances de l'installation : Bash, Binutils, Bison, Coreutils, Flex, GCC, Gettext, Glibc, Gzip, Make, Patch et Sed

Dépendances de la suite de tests : Pas de suite de tests disponible

Doit être installée préalablement : Aucune

Dépendances facultatives: Aucune

Kmod

Dépendances de l'installation : Bash, Binutils, Bison, Coreutils, Flex, GCC, Gettext, Glibc, Gzip, Make, Sed, Xz-Utils, Zlib

Dépendances de la suite de tests : Pas de suite de tests disponible

Doit être installée préalablement : Udev

Dépendances facultatives: Aucune

Less

Dépendances de l'installation : Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Grep, Make, Ncurses et Sed

Dépendances de la suite de tests : Pas de suite de tests disponible

Doit être installée préalablement : Gzip

Dépendances facultatives: Pcre

Libpipeline

Dépendances de l'installation : Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Make, Sed et Texinfo

Dépendances de la suite de tests : Pas de suite de tests disponible

Doit être installée préalablement : Man-DB

Dépendances facultatives: Aucun

Libtool

Dépendances de l'installation : Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Make, Sed et Texinfo

Dépendances de la suite de tests : Findutils

Doit être installée préalablement : Aucune

Dépendances facultatives: Aucune

Linux Kernel

Dépendances de l'installation : Bash, Binutils, Coreutils, Diffutils, Findutils, GCC, Glibc, Grep, Gzip, Kmod, Make, Ncurses, Perl et Sed

Dépendances de la suite de tests : Pas de suite de tests disponible

Doit être installée préalablement : Aucune

Dépendances facultatives: Aucune

M4

Dépendances de l'installation : Bash, Binutils, Coreutils, GCC, Glibc, Grep, Make, Sed et Texinfo

Dépendances de la suite de tests : Diffutils

Doit être installée préalablement : Autoconf et Bison

Dépendances facultatives: libsigsegv

Make

Dépendances de l'installation : Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, Make, Sed et Texinfo

Dépendances de la suite de tests : Perl et procps-ng

Doit être installée préalablement : Aucune

Dépendances facultatives: Aucune

Man-DB

Dépendances de l'installation : Bash, Binutils, Bzip2, Coreutils, Flex, GCC, GDBM, Gettext, Glibc, Grep, Groff, Gzip, Less, Libpipeline, Make, Sed et Xz

Dépendances de la suite de tests : Non lancé. Exige les paquets de suite de tests de Man-DB

Doit être installée préalablement : Aucune

Dépendances facultatives: Aucune

Man-Pages

Dépendances de l'installation : Bash, Coreutils et Make

Dépendances de la suite de tests : Pas de suite de tests disponible

Doit être installée préalablement : Aucune

Dépendances facultatives: Aucune

MPC

Dépendances de l'installation : Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, GMP, Make, MPFR, Sed et Texinfo

Dépendances de la suite de tests : Aucune

Doit être installée préalablement : GCC

Dépendances facultatives: Aucun

MPFR

Dépendances de l'installation : Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, GMP, Make, Sed et Texinfo

Dépendances de la suite de tests : Aucune

Doit être installée préalablement : GCC

Dépendances facultatives: Aucune

Ncurses

Dépendances de l'installation : Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Make, Patch et Sed

Dépendances de la suite de tests : Pas de suite de tests disponible

Doit être installée préalablement : Bash, GRUB, Inetutils, Less, procps-ng, Psmisc, Readline, Texinfo, Util-linux et Vim

Dépendances facultatives: Aucune

Patch

Dépendances de l'installation : Bash, Binutils, Coreutils, GCC, Glibc, Grep, Make et Sed

Dépendances de la suite de tests : Pas de suite de tests disponible

Doit être installée préalablement : Aucune

Dépendances facultatives: Ed

Perl

Dépendances de l'installation : Bash, Binutils, Coreutils, Gawk, GCC, GDBM, Glibc, Grep, Groff, Make, Sed et Zlib

Dépendances de la suite de tests : Iana-Etc et procps-ng

Doit être installée préalablement : Autoconf

Dépendances facultatives: Aucune

Pkg-config

Dépendances de l'installation : Bash, Binutils, Coreutils, Gawk, GCC, Glibc, Grep, Make, Popt et Sed

Dépendances de la suite de tests : Aucun

Doit être installée préalablement : Kmod

Dépendances facultatives: Aucun

Popt

Dépendances de l'installation : Bash, Binutils, Coreutils, Gawk, GCC, Glibc, Grep, Make

Dépendances de la suite de tests : Diffutils et Sed

Doit être installée préalablement : Pkg-config

Dépendances facultatives: Aucun

procps-ng

Dépendances de l'installation : Bash, Binutils, Coreutils, GCC, Glibc, Make et Ncurses

Dépendances de la suite de tests : DejaGNU

Doit être installée préalablement : Aucune

Dépendances facultatives: Aucune

Psmisc

Dépendances de l'installation : Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, Make, Ncurses et Sed

Dépendances de la suite de tests : Pas de suite de tests disponible

Doit être installée préalablement : Aucune

Dépendances facultatives: Aucune

Readline

Dépendances de l'installation :	Bash, Binutils, Coreutils, Gawk, GCC, Glibc, Grep, Make, Ncurses, Patch, Sed et Texinfo
Dépendances de la suite de tests :	Pas de suite de tests disponible
Doit être installée préalablement :	Bash
Dépendances facultatives:	Aucune

Sed

Dépendances de l'installation :	Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, Make, Sed et Texinfo
Dépendances de la suite de tests :	Diffutils et Gawk
Doit être installée préalablement :	E2fsprogs, File, Libtool et Shadow
Dépendances facultatives:	Cracklib

Shadow

Dépendances de l'installation :	Bash, Binutils, Coreutils, Diffutils, Findutils, Gawk, GCC, Gettext, Glibc, Grep, Make et Sed
Dépendances de la suite de tests :	Pas de suite de tests disponible
Doit être installée préalablement :	Coreutils
Dépendances facultatives:	Acl, Attr, Cracklib, PAM

Sysklogd

Dépendances de l'installation :	Binutils, Coreutils, GCC, Glibc, Make et Patch
Dépendances de la suite de tests :	Pas de suite de tests disponible
Doit être installée préalablement :	Aucune
Dépendances facultatives:	Aucune

Sysvinit

Dépendances de l'installation : Binutils, Coreutils, GCC, Glibc, Make et Sed

Dépendances de la suite de tests : Pas de suite de tests disponible

Doit être installée préalablement : Aucune

Dépendances facultatives: Aucune

Tar

Dépendances de l'installation : Bash, Binutils, Bison, Coreutils, GCC, Gettext, Glibc, Grep, Inetutils, Make, Sed et Texinfo

Dépendances de la suite de tests : Autoconf, Diffutils, Findutils, Gawk et Gzip

Doit être installée préalablement : Aucune

Dépendances facultatives: Aucune

Tcl

Dépendances de l'installation : Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Grep, Make et Sed

Dépendances de la suite de tests : Aucune

Doit être installée préalablement : Aucune

Dépendances facultatives: Aucune

Texinfo

Dépendances de l'installation : Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, Make, Ncurses, Patch et Sed

Dépendances de la suite de tests : Aucune

Doit être installée préalablement : Aucune

Dépendances facultatives: Aucune

Udev

Dépendances de l'installation : Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Kmod, Make et Sed

Dépendances de la suite de tests : Pas de suite de tests disponible

Doit être installée préalablement : Aucune

Dépendances facultatives: Glib, Pci-Utils, Python, Systemd, USB-Utils

Util-linux

Dépendances de l'installation : Bash, Binutils, Coreutils, Diffutils, Findutils, Gawk, GCC, Gettext, Glibc, Grep, Make, Ncurses, Sed et Zlib

Dépendances de la suite de tests : Pas de suite de tests disponible

Doit être installée préalablement : Aucune

Dépendances facultatives: Aucune

Vim

Dépendances de l'installation : Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Grep, Make, Ncurses et Sed

Dépendances de la suite de tests : Aucune

Doit être installée préalablement : Aucune

Dépendances facultatives: Xorg, GTK+2, LessTif, Python, Tcl, Ruby et GPM

Xz

Dépendances de l'installation : Bash, Binutils, Coreutils, Diffutils, GCC, Glibc et Make.

Dépendances de la suite de tests : Aucune

Doit être installée préalablement : GRUB, Kmod, Man-DB, Udev

Dépendances facultatives: Aucune

Zlib

Dépendances de l'installation :	Bash, Binutils, Coreutils, GCC, Glibc, Make et Sed
Dépendances de la suite de tests :	Aucune
Doit être installée préalablement :	File, Kmod, Perl et Util-linux
Dépendances facultatives:	Aucune

Annexe D. Scripts de démarrage et de sysconfig version-20130123

Les scripts dans cette annexe sont listés dans le répertoire où ils résident normalement. L'ordre est `/etc/rc.d/init.d`, `/etc/sysconfig`, `/etc/sysconfig/network-devices` et `/etc/sysconfig/network-devices/services`. À l'intérieur 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 et 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
# Update      : Bruce Dubbs - bdubbs@linuxfromscratch.org
#
# Version     : LFS 7.0
#
#############################################################################

. /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 files provided by\n"
   MSG="${MSG}the ${DISTRO_MINI} book, please be so kind to inform us at\n"
   MSG="${MSG}${DISTRO_CONTACT}.\n"
   log_failure_msg "${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."
      continue
   fi
```

```
   if [ ! -x ${i} ]; then
      log_warning_msg "${i} is not executable, skipping."
      continue
   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@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 /var/run/interactive ] && source /var/run/interactive

# Attempt to stop all services started by the previous runlevel,
# and killed in this runlevel
if [ "${previous}" != "N" ]; then
   for i in $(ls -v /etc/rc.d/rc${runlevel}.d/K* 2> /dev/null)
   do
      check_script_status

      suffix=${i#/etc/rc.d/rc$runlevel.d/K[0-9][0-9]}
      prev_start=/etc/rc.d/rc$previous.d/S[0-9][0-9]$suffix
```

```
      sysinit_start=/etc/rc.d/rcS.d/S[0-9][0-9]$suffix

      if [ "${runlevel}" != "0" -a "${runlevel}" != "6" ]; then
         if [ ! -f ${prev_start} -a  ! -f ${sysinit_start} ]; then
            MSG="WARNING:\n\n${i} can't be "
            MSG="${MSG}executed because it was not "
            MSG="${MSG}not started in the previous "
            MSG="${MSG}runlevel (${previous})."
            log_warning_msg "$MSG"
            continue
         fi
      fi

      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 functions in this runlevel
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$runlevel.d/S[0-9][0-9]}
      stop=/etc/rc.d/rc$runlevel.d/K[0-9][0-9]$suffix
      prev_start=/etc/rc.d/rc$previous.d/S[0-9][0-9]$suffix

      [ -f ${prev_start} -a ! -f ${stop} ] && continue
   fi

   check_script_status

   case ${runlevel} in
      0|6)
         run ${i} stop
         ;;
      *)
         run ${i} start
         ;;
   esac

   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\"" > /var/run/interactive
```

```
else
    rm -f /var/run/interactive 2> /dev/null
fi

# Copy the boot log on initial boot only
if [ "${previous}" == "N" -a  "${runlevel}" != "S" ]; then
   cat /run/var/bootlog >> /var/log/boot.log

   # Mark the end of boot
   echo "--------" >> /var/log/boot.log

   # Remove the temporary file
   rm -f /run/var/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"

## 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"

## 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}"

SUCCESS_SUFFIX="${BRACKET}[${SUCCESS}  OK  ${BRACKET}]${NORMAL}"
FAILURE_SUFFIX="${BRACKET}[${FAILURE} FAIL ${BRACKET}]${NORMAL}"
WARNING_SUFFIX="${BRACKET}[${WARNING} WARN ${BRACKET}]${NORMAL}"

BOOTLOG=/run/var/bootlog
KILLDELAY=3

# Set any user specified environment variables e.g. HEADLESS
[ -r /etc/sysconfig/rc.site ]  && . /etc/sysconfig/rc.site

################################################################################
# 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.
            rm -f "${pidfile}"

            # 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 "/var/run/${progname}.pid" ]; then
                rm -f "/var/run/${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 it's job, and evaluate kills 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 "/var/run/${progname}.pid" ]; then
            pidfile="/var/run/${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 "${base}" -a -e "/var/run/${base}.pid" ]; then
         /bin/echo -e "${WARNING}${1} is not running but" \
            "/var/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_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 failyure 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} | {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 or -{signal} or {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/functions

```
#!/bin/sh
########################################################################
# Begin boot functions
#
# Description : Run Level Control Functions
#
# Authors     : Gerard Beekmans - gerard@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
#
#               This file is only present for backward BLFS compatibility
#
########################################################################

## Environmental setup
# Setup default values for environment
umask 022
export PATH="/bin:/usr/bin:/sbin:/usr/sbin"

# Signal sent to running processes to refresh their configuration
RELOADSIG="HUP"

# Number of seconds between STOPSIG and FALLBACK when stopping processes
KILLDELAY="3"

## 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))

## Provide an echo that supports -e and -n
# If formatting is needed, $ECHO should be used
case "`echo -e -n test`" in
   -[en]*)
      ECHO=/bin/echo
      ;;
   *)
      ECHO=echo
      ;;
esac

## 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

## 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

STRING_LENGTH="0"   # the length of the current message

#*************************************************************************
# Function - boot_mesg()
#
# Purpose:      Sending information from bootup scripts to the console
#
# Inputs:       $1 is the message
#               $2 is the colorcode for the console
#
# Outputs:      Standard Output
#
# Dependencies: - sed for parsing strings.
#          - grep for counting string length.
#
# Todo:
#*************************************************************************
boot_mesg()
{
   local ECHOPARM=""

   while true
   do
```

```
      case "${1}" in
         -n)
            ECHOPARM=" -n "
            shift 1
            ;;
         -*)
            echo "Unknown Option: ${1}"
            return 1
            ;;
         *)
            break
            ;;
      esac
   done

   ## Figure out the length of what is to be printed to be used
   ## for warning messages.
   STRING_LENGTH=$((${#1} + 1))

   # Print the message to the screen
   ${ECHO} ${ECHOPARM} -e "${2}${1}"

   # Log the message
   [ -d /run/var ] || return
   ${ECHO} ${ECHOPARM} -e "${2}${1}" >> /run/var/bootlog
}

boot_mesg_flush()
{
   # Reset STRING_LENGTH for next message
   STRING_LENGTH="0"
}

echo_ok()
{
   ${ECHO} -n -e "${CURS_UP}${SET_COL}${BRACKET}[${SUCCESS}  OK  ${BRACKET}]"
   ${ECHO} -e "${NORMAL}"
   boot_mesg_flush

   [ -d /run/var ] || return
   ${ECHO} -e "[ OK ]" >> /run/var/bootlog
}

echo_failure()
{
   ${ECHO} -n -e "${CURS_UP}${SET_COL}${BRACKET}[${FAILURE} FAIL ${BRACKET}]"
   ${ECHO} -e "${NORMAL}"
    boot_mesg_flush

   [ -d /run/var ] || return
   ${ECHO} -e "[ FAIL]"  >> /run/var/bootlog
}

echo_warning()
{
   ${ECHO} -n -e "${CURS_UP}${SET_COL}${BRACKET}[${WARNING} WARN ${BRACKET}]"
   ${ECHO} -e "${NORMAL}"
```

```
   boot_mesg_flush

   [ -d /run/var ] || return
   ${ECHO} -e "[ WARN ]"  >> /run/var/bootlog
}

echo_skipped()
{
   ${ECHO} -n -e "${CURS_UP}${SET_COL}${BRACKET}[${WARNING} SKIP ${BRACKET}]"
   ${ECHO} -e "${NORMAL}"
   boot_mesg_flush

   [ -d /run/var ] || return
   ${ECHO} -e "  [ SKIP ]" >> /run/var/bootlog
}

wait_for_user()
{
   # Wait for the user by default
   [ "${HEADLESS=0}" = "0" ] && read ENTER
}

evaluate_retval()
{
   error_value="${?}"

   if [ ${error_value} = 0 ]; then
      echo_ok
   else
      echo_failure
   fi

   # This prevents the 'An Unexpected Error Has Occurred' from trivial
   # errors.
   return 0
}

print_status()
{
   if [ "${#}" = "0" ]; then
      echo "Usage: ${0} {success|warning|failure}"
      return 1
   fi

   case "${1}" in

      success)
         echo_ok
         ;;

      warning)
         # Leave this extra case in because old scripts
         # may call it this way.
         case "${2}" in
            running)
               ${ECHO} -e -n "${CURS_UP}"
               ${ECHO} -e -n "\\033[${STRING_LENGTH}G   "
```

```
                boot_mesg "Already running." ${WARNING}
                echo_warning
                ;;
             not_running)
                ${ECHO} -e -n "${CURS_UP}"
                ${ECHO} -e -n "\\033[${STRING_LENGTH}G   "
                boot_mesg "Not running." ${WARNING}
                echo_warning
                ;;
             not_available)
                ${ECHO} -e -n "${CURS_UP}"
                ${ECHO} -e -n "\\033[${STRING_LENGTH}G   "
                boot_mesg "Not available." ${WARNING}
                echo_warning
                ;;
             *)
                # This is how it is supposed to
                # be called
                echo_warning
                ;;
          esac
       ;;

       failure)
          echo_failure
       ;;

   esac

}

reloadproc()
{
   local pidfile=""
   local failure=0

   while true
   do
      case "${1}" in
         -p)
            pidfile="${2}"
            shift 2
            ;;
         -*)
            log_failure_msg "Unknown Option: ${1}"
            return 2
            ;;
         *)
            break
            ;;
      esac
   done

   if [ "${#}" -lt "1" ]; then
      log_failure_msg "Usage: reloadproc [-p pidfile] pathname"
      return 2
   fi
```

```
   # This will ensure compatibility with previous LFS Bootscripts
   if [ -n "${PIDFILE}" ]; then
      pidfile="${PIDFILE}"
   fi

   # Is the process running?
   if [ -z "${pidfile}" ]; then
      pidofproc -s "${1}"
   else
      pidofproc -s -p "${pidfile}" "${1}"
   fi

   # Warn about stale pid file
   if [ "$?" = 1 ]; then
      boot_mesg -n "Removing stale pid file: ${pidfile}. " ${WARNING}
      rm -f "${pidfile}"
   fi

   if [ -n "${pidlist}" ]; then
      for pid in ${pidlist}
      do
         kill -"${RELOADSIG}" "${pid}" || failure="1"
      done

      (exit ${failure})
      evaluate_retval

   else
      boot_mesg "Process ${1} not running." ${WARNING}
      echo_warning
   fi
}

statusproc()
{
   local pidfile=""
   local base=""
   local ret=""

   while true
   do
      case "${1}" in
         -p)
            pidfile="${2}"
            shift 2
            ;;
         -*)
            log_failure_msg "Unknown Option: ${1}"
            return 2
            ;;
         *)
            break
            ;;
      esac
   done
```

```
   if [ "${#}" != "1" ]; then
      shift 1
      log_failure_msg "Usage: statusproc [-p pidfile] pathname"
      return 2
   fi

   # Get the process basename
   base="${1##*/}"

   # This will ensure compatibility with previous LFS Bootscripts
   if [ -n "${PIDFILE}" ]; then
      pidfile="${PIDFILE}"
   fi

   # Is the process running?
   if [ -z "${pidfile}" ]; then
      pidofproc -s "${1}"
   else
      pidofproc -s -p "${pidfile}" "${1}"
   fi

   # Store the return status
   ret=$?

   if [ -n "${pidlist}" ]; then
      ${ECHO} -e "${INFO}${base} is running with Process"\
         "ID(s) ${pidlist}.${NORMAL}"
   else
      if [ -n "${base}" -a -e "/var/run/${base}.pid" ]; then
         ${ECHO} -e "${WARNING}${1} is not running but"\
            "/var/run/${base}.pid exists.${NORMAL}"
      else
         if [ -n "${pidfile}" -a -e "${pidfile}" ]; then
            ${ECHO} -e "${WARNING}${1} is not running"\
               "but ${pidfile} exists.${NORMAL}"
         else
            ${ECHO} -e "${INFO}${1} is not running.${NORMAL}"
         fi
      fi
   fi

   # Return the status from pidofproc
   return $ret
}

# The below functions are documented in the LSB-generic 2.1.0

#***********************************************************************
# Function - pidofproc [-s] [-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
#
# Outputs: return 0 - Success, pid's in stdout
#          return 1 - Program is dead, pidfile exists
```

```
#          return 2 - Invalid or excessive number of arguments,
#                     warning in stdout
#          return 3 - Program is not running
#
# Dependencies: pidof, echo, head
#
# Todo: Remove dependency on head
#       This replaces getpids
#       Test changes to pidof
#
#****************************************************************************
pidofproc()
{
   local pidfile=""
   local lpids=""
   local silent=""
   pidlist=""
   while true
   do
      case "${1}" in
         -p)
            pidfile="${2}"
            shift 2
            ;;

         -s)
            # Added for legacy opperation of getpids
            # eliminates several '> /dev/null'
            silent="1"
            shift 1
            ;;
         -*)
            log_failure_msg "Unknown Option: ${1}"
            return 2
            ;;
         *)
            break
            ;;
      esac
   done

   if [ "${#}" != "1" ]; then
      shift 1
      log_failure_msg "Usage: pidofproc [-s] [-p pidfile] pathname"
      return 2
   fi

   if [ -n "${pidfile}" ]; then
      if [ ! -r "${pidfile}" ]; then
         return 3 # Program is not running
      fi

      lpids=`head -n 1 ${pidfile}`
      for pid in ${lpids}
      do
         if [ "${pid}" -ne "$$" -a "${pid}" -ne "${PPID}" ]; then
            kill -0 "${pid}" 2>/dev/null &&
```

```
            pidlist="${pidlist} ${pid}"
         fi

         if [ "${silent}" != "1" ]; then
            echo "${pidlist}"
         fi

         test -z "${pidlist}" &&
         # Program is dead, pidfile exists
         return 1
         # else
         return 0
      done

   else
      pidlist=`pidof -o $$ -o $PPID -x "$1"`
      if [ "${silent}" != "1" ]; then
         echo "${pidlist}"
      fi

      # Get provide correct running status
      if [ -n "${pidlist}" ]; then
         return 0
      else
         return 3
      fi

   fi

   if [ "$?" != "0" ]; then
      return 3 # Program is not running
   fi
}

#****************************************************************************
# Function - loadproc [-f] [-n nicelevel] [-p pidfile] pathname [args]
#
# Purpose: This runs the specified program as a daemon
#
# Inputs: -f, run the program even if it is already running
#         -n nicelevel, specifies a nice level. See nice(1).
#         -p pidfile, uses the specified pidfile
#         pathname, pathname to the specified program
#         args, arguments to pass to specified program
#
# Outputs: return 0 - Success
#          return 2 - Invalid of excessive number of arguments,
#                     warning in stdout
#          return 4 - Program or service status is unknown
#
# Dependencies: nice, rm
#
# Todo: LSB says this should be called start_daemon
#       LSB does not say that it should call evaluate_retval
#       It checks for PIDFILE, which is deprecated.
#         Will be removed after BLFS 6.0
#       loadproc returns 0 if program is already running, not LSB compliant
```

```
#
#*****************************************************************************
loadproc()
{
   local pidfile=""
   local forcestart=""
   local nicelevel="10"

# This will ensure compatibility with previous LFS Bootscripts
   if [ -n "${PIDFILE}" ]; then
      pidfile="${PIDFILE}"
   fi

  while true
   do
      case "${1}" in
         -f)
            forcestart="1"
            shift 1
            ;;
         -n)
            nicelevel="${2}"
            shift 2
            ;;
         -p)
            pidfile="${2}"
            shift 2
            ;;
         -*)
            log_failure_msg "Unknown Option: ${1}"
            return 2 #invalid or excess argument(s)
            ;;
         *)
            break
            ;;
      esac
   done

   if [ "${#}" = "0" ]; then
      log_failure_msg "Usage: loadproc [-f] [-n nicelevel] [-p pidfile] pathname [args]"
      return 2 #invalid or excess argument(s)
   fi

   if [ -z "${forcestart}" ]; then
      if [ -z "${pidfile}" ]; then
         pidofproc -s "${1}"
      else
         pidofproc -s -p "${pidfile}" "${1}"
      fi

      case "${?}" in
         0)
            log_warning_msg "Unable to continue: ${1} is running"
            return 0 # 4
            ;;
         1)
            boot_mesg "Removing stale pid file: ${pidfile}" ${WARNING}
```

```
            rm -f "${pidfile}"
            ;;
         3)
            ;;
         *)
            log_failure_msg "Unknown error code from pidofproc: ${?}"
            return 4
            ;;
      esac
   fi

   nice -n "${nicelevel}" "${@}"
   evaluate_retval # This is "Probably" not LSB compliant,
#                  but required to be compatible with older bootscripts
   return 0
}

#****************************************************************************
# Function - killproc  [-p pidfile] pathname [signal]
#
# Purpose:
#
# Inputs: -p pidfile, uses the specified pidfile
#         pathname, pathname to the specified program
#         signal, send this signal to pathname
#
# Outputs: return 0 - Success
#          return 2 - Invalid of excessive number of arguments,
#                     warning in stdout
#          return 4 - Unknown Status
#
# Dependencies: kill, rm
#
# Todo: LSB does not say that it should call evaluate_retval
#       It checks for PIDFILE, which is deprecated.
#         Will be removed after BLFS 6.0
#
#****************************************************************************
killproc()
{
   local pidfile=""
   local killsig=TERM # default signal is SIGTERM
   pidlist=""

   # This will ensure compatibility with previous LFS Bootscripts
   if [ -n "${PIDFILE}" ]; then
      pidfile="${PIDFILE}"
   fi

   while true
   do
      case "${1}" in
         -p)
            pidfile="${2}"
            shift 2
            ;;
         -*)
```

```
            log_failure_msg "Unknown Option: ${1}"
            return 2
            ;;
         *)
            break
            ;;
      esac
   done

   if [ "${#}" = "2" ]; then
      killsig="${2}"
   elif [ "${#}" != "1" ]; then
      shift 2
      log_failure_msg "Usage: killproc  [-p pidfile] pathname [signal]"
      return 2
   fi

   # Is the process running?
   if [ -z "${pidfile}" ]; then
      pidofproc -s "${1}"
   else
      pidofproc -s -p "${pidfile}" "${1}"
   fi

   # Remove stale pidfile
   if [ "$?" = 1 ]; then
      boot_mesg "Removing stale pid file: ${pidfile}." ${WARNING}
      rm -f "${pidfile}"
   fi

    # If running, send the signal
    if [ -n "${pidlist}" ]; then
   for pid in ${pidlist}
   do
      kill -${killsig} ${pid} 2>/dev/null

      # Wait up to 3 seconds, for ${pid} to terminate
      case "${killsig}" in
      TERM|SIGTERM|KILL|SIGKILL)
         # sleep in 1/10ths of seconds and
         # multiply KILLDELAY by 10
         local dtime="${KILLDELAY}0"
         while [ "${dtime}" != "0" ]
         do
            kill -0 ${pid} 2>/dev/null || break
            sleep 0.1
            dtime=$(( ${dtime} - 1))
         done
         # If ${pid} is still running, kill it
         kill -0 ${pid} 2>/dev/null && kill -KILL ${pid} 2>/dev/null
         ;;
      esac
   done

   # Check if the process is still running if we tried to stop it
   case "${killsig}" in
   TERM|SIGTERM|KILL|SIGKILL)
```

```
      if [ -z "${pidfile}" ]; then
         pidofproc -s "${1}"
      else
         pidofproc -s -p "${pidfile}" "${1}"
      fi

      # Program was terminated
      if [ "$?" != "0" ]; then
         # Remove the pidfile if necessary
         if [ -f "${pidfile}" ]; then
            rm -f "${pidfile}"
         fi
         echo_ok
         return 0
      else # Program is still running
         echo_failure
         return 4 # Unknown Status
      fi
      ;;
   *)
      # Just see if the kill returned successfully
      evaluate_retval
      ;;
   esac
    else # process not running
   print_status warning not_running
    fi
}


#****************************************************************************
# Function - log_success_msg "message"
#
# Purpose: Print a success message
#
# Inputs: $@ - Message
#
# Outputs: Text output to screen
#
# Dependencies: echo
#
# Todo: logging
#
#****************************************************************************
log_success_msg()
{
   ${ECHO} -n -e "${BOOTMESG_PREFIX}${@}"
   ${ECHO} -e "${SET_COL}""${BRACKET}""[""${SUCCESS}""  OK  ""${BRACKET}""]""${NORMAL}"

   [ -d /run/var ] || return 0
   ${ECHO} -n -e "${@}  [ OK ]"  >> /run/var/bootlog
   return 0
}

#****************************************************************************
# Function - log_failure_msg "message"
#
```

```
# Purpose: Print a failure message
#
# Inputs: $@ - Message
#
# Outputs: Text output to screen
#
# Dependencies: echo
#
# Todo: logging
#
#*****************************************************************************
log_failure_msg() {
   ${ECHO} -n -e "${BOOTMESG_PREFIX}${@}"
   ${ECHO} -e "${SET_COL}""${BRACKET}""["${FAILURE}"" FAIL ""${BRACKET}""]""${NORMAL}"

   [ -d /run/var ] || return 0
   ${ECHO} -e "${@}  [ FAIL ]" >> /run/var/bootlog
   return 0
}

#*****************************************************************************
# Function - log_warning_msg "message"
#
# Purpose: print a warning message
#
# Inputs: $@ - Message
#
# Outputs: Text output to screen
#
# Dependencies: echo
#
# Todo: logging
#
#*****************************************************************************
log_warning_msg() {
   ${ECHO} -n -e "${BOOTMESG_PREFIX}${@}"
   ${ECHO} -e "${SET_COL}""${BRACKET}""["${WARNING}"" WARN ""${BRACKET}""]""${NORMAL}"

   [ -d /run/var ] || return 0
   ${ECHO} -e "${@}  [ WARN ]" >> /run/var/bootlog
   return 0
}

#*****************************************************************************
# Function - log_skipped_msg "message"
#
# Purpose: print a message that the script was skipped
#
# Inputs: $@ - Message
#
# Outputs: Text output to screen
#
# Dependencies: echo
#
# Todo: logging
#
#*****************************************************************************
```

```
log_skipped_msg() {
   ${ECHO} -n -e "${BOOTMESG_PREFIX}${@}"
   ${ECHO} -e "${SET_COL}""${BRACKET}""["${WARNING}" SKIP "${BRACKET}"]""${NORMAL}"

   [ -d /run/var ] || return 0
   ${ECHO} -e "${@}  [ SKIP ]" >> /run/var/bootlog
   return 0
}

# End boot functions
```

D.4. /etc/rc.d/init.d/mountvirtfs

```
#!/bin/sh
########################################################################
# Begin mountvirtfs
#
# Description : Mount proc, sysfs, and run
#
# 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:
# Should-Start:
# Required-Stop:
# Should-Stop:
# Default-Start:       S
# Default-Stop:
# Short-Description:   Mounts /sys and /proc virtual (kernel) filesystems.
#                      Mounts /run (tmpfs) and /dev (devtmpfs).
# Description:         Mounts /sys and /proc virtual (kernel) filesystems.
#                      Mounts /run (tmpfs) and /dev (devtmpfs).
# X-LFS-Provided-By:   LFS
### END INIT INFO

. /lib/lsb/init-functions

case "${1}" in
   start)
      # Make sure /run/var is available before logging any messages
      if ! mountpoint /run >/dev/null; then
         mount -n /run || failed=1
      fi

      mkdir -p /run/var /run/lock /run/shm
      chmod 1777 /run/shm

      log_info_msg "Mounting virtual file systems: ${INFO}/run"
```

```
      if ! mountpoint /proc >/dev/null; then
         log_info_msg2 " ${INFO}/proc"
         mount -n -o nosuid,noexec,nodev /proc || failed=1
      fi

      if ! mountpoint /sys >/dev/null; then
         log_info_msg2 " ${INFO}/sys"
         mount -n -o nosuid,noexec,nodev /sys || failed=1
      fi

      if ! mountpoint /dev >/dev/null; then
         log_info_msg2 " ${INFO}/dev"
         mount -n -o mode=0755,nosuid /dev  || failed=1
      fi

      # Copy devices that Udev >= 155 doesn't handle to /dev
      cp -a /lib/udev/devices/* /dev

      ln -sfn /run/shm /dev/shm

      (exit ${failed})
      evaluate_retval
      exit $failed
      ;;

   *)
      echo "Usage: ${0} {start}"
      exit 1
      ;;
esac

# End mountvirtfs
```

D.5. /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 sysctl
# 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/ksyms -o -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.6. /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:
# 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

      # Udev handles uevents itself, so we don't need to have
      # the kernel call out to any binary in response to them
      echo > /proc/sys/kernel/hotplug
```

```
      # Start the udev daemon to continually watch for, and act on,
      # uevents
      /lib/udev/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.7. /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:
# 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.8. /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 BLFS
### 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.9. /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 $time
# 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="\n/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.  This 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 [ "${error_value}" -ge 16 ]; then
         msg="\nFAILURE:\n\nUnexpected Failure "
         msg="${msg}running fsck.  Exited with error "
         msg="${msg} code: ${error_value}."
         log_failure_msg $msg
         exit ${error_value}
      fi

      exit 0
      ;;
   *)
      echo "Usage: ${0} {start}"
      exit 1
      ;;
esac

# End checkfs
```

D.10. /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:
# Required-Stop:       swap
# 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 -n -o remount,rw / >/dev/null
      evaluate_retval

      # Remove fsck-related file system watermarks.
      rm -f /fastboot /forcefsck

      log_info_msg "Recording existing mounts in /etc/mtab..."
      > /etc/mtab

      mount -f /     || failed=1
      mount -f /proc || failed=1
      mount -f /sys  || failed=1
      mount -f /run  || failed=1
      mount -f /dev  || failed=1
      (exit ${failed})
      evaluate_retval

      # 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..."
      mount -a -O no_netdev >/dev/null
      evaluate_retval
      exit $failed
      ;;

   stop)
      # Don't unmount tmpfs like /run
      log_info_msg "Unmounting all other currently mounted file systems..."
      umount -a -d -r -t notmpfs,nosysfs,nodevtmpfs,noproc >/dev/null
      evaluate_retval

      # 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.11. /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
# 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.12. /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 /var/run, /var/lock, and
#                      optionally, /tmp.  cleanfs also creates /var/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

      > /var/run/utmp

      if grep -q '^utmp:' /etc/group ; then
         chmod 664 /var/run/utmp
         chgrp utmp /var/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.13. /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:
# Should-Start:        $local_fs
# 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

is_true()
{
   [ "$1" = "1" ] || [ "$1" = "yes" ] || [ "$1" = "true" ]
}

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.14. /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:      $local_fs
# Should-Start:
# 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

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.15. /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:
# 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.16. /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:      localnet
# Should-Start:
# Required-Stop:       $local_fs sendsignals
# Should-Stop:
# 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.17. /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 swap localnet
# Should-Start:        $syslog
# Required-Stop:       $local_fs swap localnet
# Should-Stop:         $syslog
# Default-Start:       3 4 5
# Default-Stop:        0 1 2 6
# Short-Description:   Starts and configures network interfaces.
# Description:         Starts and configures network interfaces.
# X-LFS-Provided-By:   LFS
### END INIT INFO
```

```
case "${1}" in
   start)
      # Start all network interfaces
      for file in /etc/sysconfig/ifconfig.*
      do
         interface=${file##*/ifconfig.}

         # Skip if $file is * (because nothing was found)
         if [ "${interface}" = "*" ]
         then
            continue
         fi

         /sbin/ifup ${interface}
      done
      ;;

   stop)
      # 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

         /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.18. /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 "${1}" in
   stop)
      log_info_msg "Sending all processes the TERM signal..."
      killall5 -15
      error_value=${?}

      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
      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.19. /etc/rc.d/init.d/reboot

```
#!/bin/sh
########################################################################
# Begin reboot
#
# Description : Reboot Scripts
#
# 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:            reboot
# Required-Start:
# 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
   stop)
      log_info_msg "Restarting system..."
      reboot -d -f -i
      ;;

   *)
      echo "Usage: ${0} {stop}"
      exit 1
      ;;

esac
```

```
# End reboot
```

D.20. /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
#
# Version     : LFS 7.0
#
########################################################################

### BEGIN INIT INFO
# Provides:            halt
# Required-Start:
# 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
   stop)
      halt -d -f -i -p
      ;;

   *)
      echo "Usage: {stop}"
      exit 1
      ;;
esac

# End halt
```

D.21. /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..."
      start_daemon fully_qualified_path
      ;;

   stop)
      log_info_msg "Stopping..."
      killproc fully_qualified_path
      ;;

   restart)
      ${0} stop
      sleep 1
      ${0} start
      ;;

   *)
      echo "Usage: ${0} {start|stop|restart}"
      exit 1
      ;;
esac

exit 0

# End scriptname
```

D.22. /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 it's 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.23. /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.24. /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.25. /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
#
# Version     : LFS 7.2
#
# 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()
{
  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 "\nInterface ${IFACE} doesn't exist."
     exit 1
  fi
}

RELEASE="7.2"

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

log_info_msg "Bringing up the ${1} interface... "

if [ ! -r "${file}" ]; then
   log_failure_msg2 "${file} is missing or cannot be accessed."
   exit 1
fi

. $file

if [ "$IFACE" = "" ]; then
   log_failure_msg2 "${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
   log_info_msg2 "skipped"
   exit 0
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

# Create/configure the interface
for S in ${SERVICE}; do
  IFCONFIG=${file} /lib/services/${S} ${IFACE} up
done

# Bring up the interface and any components
for I in $IFACE $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 "\nGateway already setup; skipping."
   else
      log_info_msg "Setting up default gateway..."
      ip route add default via ${GATEWAY} dev ${IFACE}
      evaluate_retval
   fi
fi

# End /sbin/ifup
```

D.26. /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.27. /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 "${BROADCAST}" ]; then
   args="${args} broadcast ${BROADCAST}"
fi

case "${2}" in
   up)
      if [ "$(ip addr show ${1} 2>/dev/null | grep ${IP}/)" = "" ]; then

         # Cosmetic output not needed for multiple services
         if ! $(echo ${SERVICE} | grep -q " "); then
           log_info_msg2 "\n" # Terminate the previous message
         fi

         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.28. /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 "${need_ip}" ]; then
   if [ -z "${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 "${need_gateway}" ]; then
   if [ -z "${GATEWAY}" ]; then
      log_failure_msg "GATEWAY variable missing from ${IFCONFIG}, cannot continue."
      exit 1
   fi
   args="${args} via ${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 tirées de udev-lfs-197-2.tar.bz2 sont énumérées dans cette annexe par soucis de commodité. L'installation se fait d'ordinaire en suivant les instructions de Section 6.61, « Udev-197 (Extrait de systemd-197) ».

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 License.

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 © 1999-2012 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

Autoconf: 155
Automake: 157
Bash: 146
  outils: 56
Binutils: 101
  outils, passe 1: 34
  outils, passe 2: 43
Bison: 142
Bootscripts: 214
  utilisation: 216
Bzip2: 115
  outils: 57
Check: 54
Coreutils: 135
  outils: 58
DejaGNU: 53
Diffutils: 159
  outils: 59
E2fsprogs: 128
Expect: 51
File: 100
  outils: 60
Findutils: 161
  outils: 61
Flex: 163
Gawk: 160
  outils: 62
GCC: 108
  outils, passe 1: 36
  outils, passe 2: 45
GDBM: 149
Gettext: 165
  outils: 63
Glibc: 89
  outils: 40
GMP: 104
Grep: 143
  outils: 64
Groff: 167
GRUB: 172
Gzip: 175
  outils: 65
Iana-Etc: 140
Inetutils: 150
IPRoute2: 177
Kbd: 179
Kmod: 182
Less: 174
Libpipeline: 184
Libtool: 148
Linux: 232
  en-têtes API: 86
  outils, en-têtes API: 39
M4: 141
  outils: 66
Make: 185
  outils: 67
Man-DB: 186
Man-pages: 88
MPC: 107
MPFR: 106
Ncurses: 118
  outils: 55
Patch: 189
  outils: 68
Perl: 152
  outils: 69
pkgconfig: 117
Procps-ng: 126
Psmisc: 125
rc.site: 223
Readline: 144
Sed: 114
  outils: 70
Shadow: 131
  configuration: 132
Sysklogd: 190
  configuration: 190
Sysvinit: 191
  configuration: 217
Tar: 193
  outils: 71
Tcl: 49
Texinfo: 194
  outils: 72
Udev: 196
  utilisation: 207
Util-linux: 121
Vim: 198
xz: 170

tools: 73
Zlib: 99

Programmes

a2p: 152, 153
accessdb: 186, 187
acinstall: 157, 157
aclocal: 157, 157
aclocal-1.13: 157, 157
addftinfo: 167, 167
addpart: 121, 122
addr2line: 101, 102
afmtodit: 167, 167
agetty: 121, 122
apropos: 186, 188
ar: 101, 102
as: 101, 102
ata_id: 196, 197
autoconf: 155, 155
autoheader: 155, 155
autom4te: 155, 155
automake: 157, 157
automake-1.13: 157, 157
autopoint: 165, 165
autoreconf: 155, 155
autoscan: 155, 155
autoupdate: 155, 155
awk: 160, 160
badblocks: 128, 129
base64: 135, 136
basename: 135, 136
bash: 146, 147
bashbug: 146, 147
bigram: 161, 161
bison: 142, 142
blkid: 121, 122
blockdev: 121, 122
bootlogd: 191, 191
bridge: 177, 177
bunzip2: 115, 116
bzcat: 115, 116
bzcmp: 115, 116
bzdiff: 115, 116
bzegrep: 115, 116
bzfgrep: 115, 116
bzgrep: 115, 116
bzip2: 115, 116
bzip2recover: 115, 116
bzless: 115, 116
bzmore: 115, 116
c++: 108, 112
c++filt: 101, 102
c2ph: 152, 153
cal: 121, 122
captoinfo: 118, 120
cat: 135, 136
catchsegv: 89, 94
catman: 186, 188
cc: 108, 112
cdrom_id: 196, 197
cfdisk: 121, 122
chage: 131, 133
chattr: 128, 129
chcon: 135, 136
chcpu: 121, 122
checkmk: 54, 54
chem: 167, 168
chfn: 131, 133
chgpasswd: 131, 133
chgrp: 135, 136
chmod: 135, 136
chown: 135, 137
chpasswd: 131, 133
chroot: 135, 137
chrt: 121, 122
chsh: 131, 133
chvt: 179, 180
cksum: 135, 137
clear: 118, 120
cmp: 159, 159
code: 161, 161
col: 121, 122
colcrt: 121, 122
collect: 196, 197
colrm: 121, 122
column: 121, 122
comm: 135, 137
compile: 157, 157
compile_et: 128, 129
config.charset: 165, 165
config.guess: 157, 158
config.rpath: 165, 165
config.sub: 157, 158
config_data: 152, 153

corelist: 152, 153
cp: 135, 137
cpan: 152, 153
cpan2dist: 152, 153
cpanp: 152, 153
cpanp-run-perl: 152, 153
cpp: 108, 112
csplit: 135, 137
ctrlaltdel: 121, 122
ctstat: 177, 178
cut: 135, 137
cytune: 121, 122
date: 135, 137
dd: 135, 137
deallocvt: 179, 180
debugfs: 128, 129
delpart: 121, 122
depcomp: 157, 158
depmod: 182, 183
df: 135, 137
dgawk: 160, 160
diff: 159, 159
diff3: 159, 159
dir: 135, 137
dircolors: 135, 137
dirname: 135, 137
dmesg: 121, 122
du: 135, 137
dumpe2fs: 128, 129
dumpkeys: 179, 180
e2freefrag: 128, 129
e2fsck: 128, 129
e2image: 128, 130
e2initrd_helper: 128, 130
e2label: 128, 130
e2undo: 128, 130
e4defrag: 128, 130
echo: 135, 137
egrep: 143, 143
eject: 121, 122
elfedit: 101, 102
elisp-comp: 157, 158
enc2xs: 152, 153
env: 135, 137
envsubst: 165, 165
eqn: 167, 168
eqn2graph: 167, 168
ex: 198, 200
expand: 135, 137
expect: 51, 52
expiry: 131, 133
expr: 135, 137
factor: 135, 137
faillog: 131, 133
fallocate: 121, 122
false: 135, 137
fdformat: 121, 122
fdisk: 121, 122
fgconsole: 179, 180
fgrep: 143, 143
file: 100, 100
filefrag: 128, 130
find: 161, 162
find2perl: 152, 153
findfs: 121, 122
findmnt: 121, 122
flex: 163, 164
flex++: 163, 164
flock: 121, 122
fmt: 135, 137
fold: 135, 137
frcode: 161, 162
free: 126, 127
fsck: 121, 122
fsck.cramfs: 121, 122
fsck.ext2: 128, 130
fsck.ext3: 128, 130
fsck.ext4: 128, 130
fsck.ext4dev: 128, 130
fsck.minix: 121, 122
fsfreeze: 121, 122
fstab-decode: 191, 191
fstrim: 121, 122
ftp: 150, 151
fuser: 125, 125
g++: 108, 112
gawk: 160, 160
gawk-4.0.2: 160, 160
gcc: 108, 112
gc-ar: 108, 112
gc-nm: 108, 112
gc-ranlib: 108, 112
gccbug: 108, 112
gcov: 108, 112

gdiffmk: 167, 168
gencat: 89, 94
genl: 177, 178
geqn: 167, 168
getconf: 89, 94
getent: 89, 94
getkeycodes: 179, 180
getopt: 121, 122
gettext: 165, 165
gettext.sh: 165, 165
gettextize: 165, 166
gpasswd: 131, 133
gprof: 101, 102
grap2graph: 167, 168
grcat: 160, 160
grep: 143, 143
grn: 167, 168
grodvi: 167, 168
groff: 167, 168
groffer: 167, 168
grog: 167, 168
grolbp: 167, 168
grolj4: 167, 168
grops: 167, 168
grotty: 167, 168
groupadd: 131, 133
groupdel: 131, 133
groupmems: 131, 133
groupmod: 131, 133
groups: 135, 137
grpck: 131, 133
grpconv: 131, 133
grpunconv: 131, 133
grub-bios-setup: 172, 172
grub-editenv: 172, 172
grub-fstest: 172, 172
grub-install: 172, 173
grub-kbdcomp: 172, 173
grub-menulst2cfg: 172, 173
grub-mkconfig: 172, 173
grub-mkimage: 172, 173
grub-mklayout: 172, 173
grub-mknetdir: 172, 173
grub-mkpasswd-pbkdf2: 172, 173
grub-mkrelpath: 172, 173
grub-mkrescue: 172, 173
grub-mkstandalone: 172, 173
grub-ofpathname: 172, 173
grub-probe: 172, 173
grub-reboot: 172, 173
grub-script-check: 172, 173
grub-set-default: 172, 173
grub-setup: 172, 173
gtbl: 167, 168
gunzip: 175, 175
gzexe: 175, 175
gzip: 175, 175
h2ph: 152, 153
h2xs: 152, 153
halt: 191, 191
head: 135, 137
hexdump: 121, 122
hostid: 135, 137
hostname: 150, 151
hostname: 165, 166
hpftodit: 167, 168
hwclock: 121, 122
i386: 121, 123
iconv: 89, 94
iconvconfig: 89, 94
id: 135, 137
ifcfg: 177, 178
ifnames: 155, 156
ifstat: 177, 178
igawk: 160, 160
indxbib: 167, 168
info: 194, 195
infocmp: 118, 120
infokey: 194, 195
infotocap: 118, 120
init: 191, 191
insmod: 182, 183
install: 135, 137
install-info: 194, 195
install-sh: 157, 158
instmodsh: 152, 153
ionice: 121, 123
ip: 177, 178
ipcmk: 121, 123
ipcrm: 121, 123
ipcs: 121, 123
isosize: 121, 123
join: 135, 137
json_pp: 152, 153

kbdinfo: 179, 180
kbdrate: 179, 180
kbd_mode: 179, 180
kill: 121, 123
killall: 125, 125
killall5: 191, 191
klogd: 190, 190
kmod: 182, 183
last: 191, 192
lastb: 191, 192
lastlog: 131, 133
ld: 101, 102
ld.bfd: 101, 102
ldattach: 121, 123
ldconfig: 89, 94
ldd: 89, 94
lddlibc4: 89, 94
less: 174, 174
lessecho: 174, 174
lesskey: 174, 174
lex: 163, 164
lexgrog: 186, 188
lfskernel-3.8.1: 232, 234
libnetcfg: 152, 153
libtool: 148, 148
libtoolize: 148, 148
link: 135, 137
linux32: 121, 123
linux64: 121, 123
lkbib: 167, 168
ln: 135, 137
lnstat: 177, 178
loadkeys: 179, 180
loadunimap: 179, 180
locale: 89, 94
localedef: 89, 94
locate: 161, 162
logger: 121, 123
login: 131, 133
logname: 135, 138
logoutd: 131, 133
logsave: 128, 130
look: 121, 123
lookbib: 167, 168
losetup: 121, 123
ls: 135, 138
lsattr: 128, 130
lsblk: 121, 123
lscpu: 121, 123
lslocks: 121, 123
lsmod: 182, 183
lzcat: 170, 170
lzcmp: 170, 170
lzdiff: 170, 170
lzegrep: 170, 170
lzfgrep: 170, 170
lzgrep: 170, 170
lzless: 170, 170
lzma: 170, 170
lzmadec: 170, 170
lzmainfo: 170, 171
lzmore: 170, 171
m4: 141, 141
make: 185, 185
makedb: 89, 95
makeinfo: 194, 195
man: 186, 188
mandb: 186, 188
manpath: 186, 188
mapscrn: 179, 180
mcookie: 121, 123
md5sum: 135, 138
mdate-sh: 157, 158
mesg: 191, 192
missing: 157, 158
mkdir: 135, 138
mke2fs: 128, 130
mkfifo: 135, 138
mkfs: 121, 123
mkfs.bfs: 121, 123
mkfs.cramfs: 121, 123
mkfs.ext2: 128, 130
mkfs.ext3: 128, 130
mkfs.ext4: 128, 130
mkfs.ext4dev: 128, 130
mkfs.minix: 121, 123
mkinstalldirs: 157, 158
mklost+found: 128, 130
mknod: 135, 138
mkswap: 121, 123
mktemp: 135, 138
mk_cmds: 128, 130
mmroff: 167, 168
modinfo: 182, 183

modprobe: 182, 183
more: 121, 123
mount: 121, 123
mountpoint: 121, 123
msgattrib: 165, 166
msgcat: 165, 166
msgcmp: 165, 166
msgcomm: 165, 166
msgconv: 165, 166
msgen: 165, 166
msgexec: 165, 166
msgfilter: 165, 166
msgfmt: 165, 166
msggrep: 165, 166
msginit: 165, 166
msgmerge: 165, 166
msgunfmt: 165, 166
msguniq: 165, 166
mtrace: 89, 95
mv: 135, 138
namei: 121, 123
ncurses5-config: 118, 120
neqn: 167, 168
newgrp: 131, 133
newusers: 131, 133
ngettext: 165, 166
nice: 135, 138
nl: 135, 138
nm: 101, 102
nohup: 135, 138
nologin: 131, 133
nproc: 135, 138
nroff: 167, 168
nscd: 89, 95
nstat: 177, 178
objcopy: 101, 103
objdump: 101, 103
od: 135, 138
oldfind: 161, 162
openvt: 179, 180
partx: 121, 123
passwd: 131, 133
paste: 135, 138
patch: 189, 189
pathchk: 135, 138
pcprofiledump: 89, 95
pdfroff: 167, 168
pdftexi2dvi: 194, 195
peekfd: 125, 125
perl: 152, 153
perl5.16.2: 152, 153
perlbug: 152, 154
perldoc: 152, 154
perlivp: 152, 154
perlthanks: 152, 154
pfbtops: 167, 168
pg: 121, 123
pgawk: 160, 160
pgawk-4.0.2: 160, 160
pgrep: 126, 127
pic: 167, 168
pic2graph: 167, 169
piconv: 152, 154
pidof: 191, 192
ping: 150, 151
ping6: 150, 151
pinky: 135, 138
pivot_root: 121, 123
pkg-config: 117, 117
pkill: 126, 127
pl2pm: 152, 154
pldd: 89, 95
pmap: 126, 127
pod2html: 152, 154
pod2latex: 152, 154
pod2man: 152, 154
pod2texi: 194, 195
pod2text: 152, 154
pod2usage: 152, 154
podchecker: 152, 154
podselect: 152, 154
post-grohtml: 167, 169
poweroff: 191, 192
pr: 135, 138
pre-grohtml: 167, 169
preconv: 167, 169
printenv: 135, 138
printf: 135, 138
prlimit: 121, 123
prove: 152, 154
prtstat: 125, 125
ps: 126, 127
psed: 152, 154
psfaddtable: 179, 180

psfgettable: 179, 180
psfstriptable: 179, 180
psfxtable: 179, 181
pstree: 125, 125
pstree.x11: 125, 125
pstruct: 152, 154
ptar: 152, 154
ptardiff: 152, 154
ptargrep: 152, 154
ptx: 135, 138
pt_chown: 89, 95
pwcat: 160, 160
pwck: 131, 133
pwconv: 131, 133
pwd: 135, 138
pwdx: 126, 127
pwunconv: 131, 133
py-compile: 157, 158
ranlib: 101, 103
raw: 121, 123
rcp: 150, 151
readelf: 101, 103
readlink: 135, 138
readprofile: 121, 123
realpath: 135, 138
reboot: 191, 192
recode-sr-latin: 165, 166
refer: 167, 169
rename: 121, 123
renice: 121, 123
reset: 118, 120
resize2fs: 128, 130
resizepart: 121, 124
rev: 121, 124
rexec: 150, 151
rlogin: 150, 151
rm: 135, 138
rmdir: 135, 138
rmmod: 182, 183
rmt: 193, 193
roff2dvi: 167, 169
roff2html: 167, 169
roff2pdf: 167, 169
roff2ps: 167, 169
roff2text: 167, 169
roff2x: 167, 169
routef: 177, 178
routel: 177, 178
rpcgen: 89, 95
rsh: 150, 151
rtacct: 177, 178
rtcwake: 121, 124
rtmon: 177, 178
rtpr: 177, 178
rtstat: 177, 178
runcon: 135, 138
runlevel: 191, 192
runtest: 53, 53
rview: 198, 200
rvim: 198, 200
s2p: 152, 154
script: 121, 124
scriptreplay: 121, 124
scsi_id: 196, 197
sdiff: 159, 159
sed: 114, 114
seq: 135, 138
setarch: 121, 124
setfont: 179, 181
setkeycodes: 179, 181
setleds: 179, 181
setmetamode: 179, 181
setsid: 121, 124
setterm: 121, 124
sfdisk: 121, 124
sg: 131, 133
sh: 146, 147
sha1sum: 135, 138
sha224sum: 135, 138
sha256sum: 135, 138
sha384sum: 135, 138
sha512sum: 135, 138
shasum: 152, 154
showconsolefont: 179, 181
showkey: 179, 181
shred: 135, 139
shuf: 135, 139
shutdown: 191, 192
size: 101, 103
slabtop: 126, 127
sleep: 135, 139
sln: 89, 95
soelim: 167, 169
sort: 135, 139

sotruss: 89, 95
splain: 152, 154
split: 135, 139
sprof: 89, 95
ss: 177, 178
stat: 135, 139
stdbuf: 135, 139
strings: 101, 103
strip: 101, 103
stty: 135, 139
su: 131, 133
sulogin: 191, 192
sum: 135, 139
swaplabel: 121, 124
swapoff: 121, 124
swapon: 121, 124
switch_root: 121, 124
symlink-tree: 157, 158
sync: 135, 139
sysctl: 126, 127
syslogd: 190, 190
tac: 135, 139
tabs: 118, 120
tail: 135, 139
tailf: 121, 124
talk: 150, 151
tar: 193, 193
taskset: 121, 124
tbl: 167, 169
tc: 177, 178
tclsh: 49, 50
tclsh8.5: 49, 50
tee: 135, 139
telinit: 191, 192
telnet: 150, 151
test: 135, 139
testgdbm: 149, 149
texi2dvi: 194, 195
texi2pdf: 194, 195
texi2any: 194, 195
texindex: 194, 195
tfmtodit: 167, 169
tftp: 150, 151
tic: 118, 120
timeout: 135, 139
tload: 126, 127
toe: 118, 120
top: 126, 127
touch: 135, 139
tput: 118, 120
tr: 135, 139
traceroute: 150, 151
troff: 167, 169
true: 135, 139
truncate: 135, 139
tset: 118, 120
tsort: 135, 139
tty: 135, 139
tune2fs: 128, 130
tunelp: 121, 124
tzselect: 89, 95
udevadm: 196, 197
udevd: 196, 197
ul: 121, 124
umount: 121, 124
uname: 135, 139
uncompress: 175, 175
unexpand: 135, 139
unicode_start: 179, 181
unicode_stop: 179, 181
uniq: 135, 139
unlink: 135, 139
unlzma: 170, 171
unshare: 121, 124
unxz: 170, 171
updatedb: 161, 162
uptime: 126, 127
useradd: 131, 133
userdel: 131, 133
usermod: 131, 133
users: 135, 139
utmpdump: 121, 124
uuidd: 121, 124
uuidgen: 121, 124
vdir: 135, 139
vi: 198, 200
view: 198, 200
vigr: 131, 134
vim: 198, 200
vimdiff: 198, 200
vimtutor: 198, 200
vipw: 131, 134
vmstat: 126, 127
w: 126, 127

wall: 121, 124
watch: 126, 127
wc: 135, 139
wdctl: 121, 124
whatis: 186, 188
whereis: 121, 124
who: 135, 139
whoami: 135, 139
wipefs: 121, 124
x86_64: 121, 124
xargs: 161, 162
xgettext: 165, 166
xsubpp: 152, 154
xtrace: 89, 95
xxd: 198, 200
xz: 170, 171
xzcat: 170, 171
xzcmp: 170, 171
xzdec: 170, 171
xzdiff: 170, 171
xzegrep: 170, 171
xzfgrep: 170, 171
xzgrep: 170, 171
xzless: 170, 171
xzmore: 170, 171
yacc: 142, 142
yes: 135, 139
ylwrap: 157, 158
zcat: 175, 175
zcmp: 175, 175
zdiff: 175, 175
zdump: 89, 95
zegrep: 175, 175
zfgrep: 175, 175
zforce: 175, 175
zgrep: 175, 176
zic: 89, 95
zipdetails: 152, 154
zless: 175, 176
zmore: 175, 176
znew: 175, 176
zsoelim: 186, 188

BibliothÃ¨ques

ld.so: 89, 95
libanl: 89, 95
libasprintf: 165, 166
libbfd: 101, 103
libblkid: 121, 124
libBrokenLocale: 89, 95
libbsd-compat: 89, 95
libbz2*: 115, 116
libc: 89, 95
libcheck: 54, 54
libcidn: 89, 95
libcom_err: 128, 130
libcrypt: 89, 95
libcurses: 118, 120
libdl: 89, 95
libe2p: 128, 130
libexpect-5.45: 51, 52
libext2fs: 128, 130
libfl.a: 163, 164
libform: 118, 120
libg: 89, 95
libgcc*: 108, 112
libgcov: 108, 112
libgdbm: 149, 149
libgettextlib: 165, 166
libgettextpo: 165, 166
libgettextsrc: 165, 166
libgmp: 104, 105
libgmpxx: 104, 105
libgomp: 108, 112
libhistory: 144, 145
libiberty: 101, 103
libieee: 89, 95
libkmod: 182
libltdl: 148, 148
liblto_plugin*: 108, 112
liblzma*: 170, 171
libm: 89, 95
libmagic: 100, 100
libman: 186, 188
libmandb: 186, 188
libmcheck: 89, 95
libmemusage: 89, 96
libmenu: 118, 120
libmount: 121, 124
libmp: 104, 105
libmpc: 107, 107
libmpfr: 106, 106
libmudflap*: 108, 112
libncurses: 118, 120

libnsl: 89, 96
libnss: 89, 96
libopcodes: 101, 103
libpanel: 118, 120
libpcprofile: 89, 96
libpipeline: 184
libprocps: 126, 127
libpthread: 89, 96
libquadmath*: 108, 112
libquota: 128, 130
libreadline: 144, 145
libresolv: 89, 96
librpcsvc: 89, 96
librt: 89, 96
libSegFault: 89, 95
libss: 128, 130
libssp*: 108, 112
libstdbuf.so: 135, 139
libstdc++: 108, 113
libsupc++: 108, 113
libtcl8.5.so: 49, 50
libtclstub8.5.a: 49, 50
libthread_db: 89, 96
libudev: 196, 197
libutil: 89, 96
libuuid: 121, 124
liby.a: 142, 142
libz: 99, 99
preloadable_libintl: 165, 166

Scripts

checkfs: 214, 214
cleanfs: 214, 214
console: 214, 214
  configuration: 219
functions: 214, 214
halt: 214, 214
hostname
  configuration: 218
ifdown: 214, 214
ifup: 214, 214
localnet: 214, 214
  /etc/hosts: 206
modules: 214, 214
mountfs: 214, 214
mountkernfs: 214, 214
network: 214, 214
  /etc/hosts: 206
  configuration: 203
rc: 214, 214
reboot: 214, 215
sendsignals: 214, 215
setclock: 214, 215
  configuration: 219
static: 214, 215
swap: 214, 215
sysctl: 214, 215
sysklogd: 214, 215
  configuration: 222
template: 214, 215
udev: 214, 215
udev_retry: 214, 215

Autres

/boot/config-3.8.1: 232, 234
/boot/System.map-3.8.1: 232, 234
/dev/*: 76
/etc/fstab: 230
/etc/group: 83
/etc/hosts: 206
/etc/inittab: 217
/etc/inputrc: 228
/etc/ld.so.conf: 93
/etc/lfs-release: 237
/etc/localtime: 91
/etc/modprobe.d/usb.conf: 234
/etc/nsswitch.conf: 91
/etc/passwd: 83
/etc/profile: 226
/etc/protocols: 140
/etc/resolv.conf: 206
/etc/services: 140
/etc/syslog.conf: 190
/etc/udev: 196, 197
/etc/vimrc: 199
/usr/include/asm-generic/*.h: 86, 86
/usr/include/asm/*.h: 86, 86
/usr/include/drm/*.h: 86, 86
/usr/include/linux/*.h: 86, 86
/usr/include/mtd/*.h: 86, 86
/usr/include/rdma/*.h: 86, 86
/usr/include/scsi/*.h: 86, 86
/usr/include/sound/*.h: 86, 86
/usr/include/video/*.h: 86, 87

/usr/include/xen/*.h: 86, 87
/var/log/btmp: 83
/var/log/lastlog: 83
/var/log/wtmp: 83
/var/run/utmp: 83
man pages: 88, 88