

Linux From Scratch compilé de façon croisée

Version GIT-20130401-x86_64-Multilib

Linux From Scratch compilé de façon croisée: Version GIT-20130401-x86_64-Multilib

Copyright © 2005–2013 Joe Ciccone, Jim Gifford & Ryan Oliver

Basé sur LFS, Copyright © 1999–2013 Gerard Beekmans

Copyright © 2005–2013, Joe Ciccone, Jim Gifford, & Ryan Oliver

Tous droits réservés.

Ce produit ne peut être distribué que s'il est soumis aux termes et les conditions indiquées plus loin la Open Publication License v1.0 ou supérieur (la dernière version est actuellement disponible sur <http://www.opencontent.org/openpub/>).

Linux® est une marque déposée de Linus Torvalds.

Ce livre se base sur le livre "Linux From Scratch", qui a été écrit sous la licence suivante :

Copyright © 1999–2013, Gerard Beekmans

Tous droits réservés.

La redistribution et l'utilisation du source ou des binaires, avec ou sans modifications, sont autorisées sous réserve des conditions suivantes :

- Les redistributions quelqu'en soit la forme doivent mentionner le copyright ci-dessus, cette liste de conditions et le dégageant de responsabilités
- Ni le nom « Linux From Scratch » ni les noms de ses contributeurs ne peuvent être utilisés à des fins commerciales ou de publicité, dérivés de cet ouvrage, sans autorisation préalable écrite
- Tout produit dérivé de Linux From Scratch doit contenir une référence au projet « Linux From Scratch »

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS « AS IS » AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Table des matières

Préface	ix
i. Avant-propos	ix
ii. Public visé	ix
iii. Prérequis	x
iv. Prérequis du système hôte	xi
v. Typographie	xii
vi. Structure	xiii
vii. Errata	xiv
I. Introduction	1
1. Introduction	2
1.1. Remerciements LFS Croisé	2
1.2. Comment construire un système CLFS	3
1.3. Historique des gros changements	4
1.4. Historique des modifications pour x86_64	11
1.5. Ressources	12
1.6. Aide	13
II. Préparation de la construction	15
2. Préparez une nouvelle partition	16
2.1. Introduction	16
2.2. Créer une nouvelle partition	16
2.3. Créer un système de fichiers sur la partition	16
2.4. Monter la nouvelle partition	17
3. Paquets et correctifs	19
3.1. Introduction	19
3.2. Tous les paquets	19
3.3. Paquets supplémentaires pour x86_64 Multilib	25
3.4. Correctifs nécessaires	25
3.5. Correctifs supplémentaires pour x86_64 multilib	27
4. Dernières préparations	28
4.1. À propos de <code>{CLFS}</code>	28
4.2. Créer le répertoire <code>{CLFS}/tools</code>	28
4.3. Créer le répertoire <code>{CLFS}/cross-tools</code>	29
4.4. Ajouter l'utilisateur CLFS	29
4.5. Configurer l'environnement	30
4.6. À propos des suites de tests	31
III. Fabriquer les outils de compilation croisée	32
5. Construire les outils de compilation croisée	33
5.1. Introduction	33
5.2. CFLAGS de construction	33
5.3. Variables de construction	34
5.4. Options de compilation	34
5.5. Linux-Headers-3.4.17	35
5.6. File-5.11	36
5.7. M4-1.4.16	37
5.8. Ncurses-5.9	38
5.9. GMP-5.0.5	39
5.10. MPFR-3.1.1	40
5.11. MPC-1.0.1	41

5.12. PPL-0.12.1	42
5.13. CLooG-0.16.3	43
5.14. Binutils-2.23 croisé	44
5.15. GCC-4.6.3 croisé - Statique	46
5.16. EGLIBC-2.15 32 bit	48
5.17. EGLIBC-2.15 64 bits	50
5.18. GCC-4.6.3 croisé - compilation finale	52
IV. Construction des outils de base	54
6. Construire un système temporaire	55
6.1. Introduction	55
6.2. Variables de construction	55
6.3. GMP-5.0.5	56
6.4. MPFR-3.1.1	57
6.5. MPC-1.0.1	58
6.6. PPL-0.12.1	59
6.7. CLooG-0.16.3	60
6.8. Zlib-1.2.7	61
6.9. Binutils-2.23	62
6.10. GCC-4.6.3	63
6.11. Ncurses-5.9	65
6.12. Bash-4.2	66
6.13. Bison-2.6.4	68
6.14. Bzip2-1.0.6	69
6.15. Coreutils-8.20	70
6.16. Diffutils-3.2	72
6.17. Findutils-4.4.2	73
6.18. File-5.11	74
6.19. Flex-2.5.37	75
6.20. Gawk-4.0.1	76
6.21. Gettext-0.18.1.1	77
6.22. Grep-2.14	78
6.23. Gzip-1.5	79
6.24. M4-1.4.16	80
6.25. Make-3.82	81
6.26. Patch-2.7.1	82
6.27. Sed-4.2.1	83
6.28. Tar-1.26	84
6.29. Texinfo-4.13a	85
6.30. Vim-7.3	86
6.31. XZ Utils-5.0.4	88
6.32. Démarrer ou se chrooter ?	89
7. Si vous allez redémarrer	90
7.1. Introduction	90
7.2. Créer les répertoires	90
7.3. Création des liens essentiels	91
7.4. Util-linux-2.22.1	92
7.5. Shadow-4.1.5.1	93
7.6. E2fsprogs-1.42.6	94
7.7. Sysvinit-2.88dsf	95
7.8. Kmod-10	97

7.9. Udev-182	98
7.10. Créer les fichiers de mot de passe, des groupes et des journaux	99
7.11. Linux-3.4.17	102
7.12. GRUB-2.00	104
7.13. Configurer l'environnement	105
7.14. Options de construction	105
7.15. Créer le fichier /etc/fstab	106
7.16. Scripts de démarrage pour CLFS 2.0-pre2	107
7.17. Peupler /dev	108
7.18. Changer de propriétaire	108
7.19. Que faire ensuite	108
8. Si vous allez vous chrooter	109
8.1. Introduction	109
8.2. Util-linux-2.22.1	110
8.3. Monter les systèmes de fichiers virtuels du noyau	111
8.4. Entrer dans l'environnement Chroot	111
8.5. Changer de propriétaire	112
8.6. Créer les répertoires	113
8.7. Créer les liens symboliques essentiels	113
8.8. Options de construction	114
8.9. Créer le mot de passe, le groupe et les fichiers journal	114
8.10. Monter les systèmes de fichiers du noyau	116
V. Construction du système CLFS	117
9. Construction des outils de test	118
9.1. Introduction	118
9.2. Tcl-8.5.12	119
9.3. Expect-5.45	120
9.4. DejaGNU-1.5	121
10. Installation des logiciels du système de base	122
10.1. Introduction	122
10.2. Gestion de paquets	122
10.3. À nouveau à propos des suites de tests	125
10.4. Perl-5.16.2 temporaire	126
10.5. Linux-Headers-3.4.17	127
10.6. Man-pages-3.43	128
10.7. EGLIBC-2.15 32 bit Libraries	129
10.8. EGLIBC-2.15 64 bits	132
10.9. Ajuster la chaîne d'outils	138
10.10. GMP-5.0.5 32 Bit Libraries	139
10.11. GMP-5.0.5 64 Bit	140
10.12. MPFR-3.1.1 32 Bit Libraries	142
10.13. MPFR-3.1.1 64 Bit	143
10.14. MPC-1.0.1 32 Bit Libraries	144
10.15. MPC-1.0.1 64 Bit	145
10.16. PPL-0.12.1 32 Bit Libraries	146
10.17. PPL-0.12.1 64 Bit	147
10.18. CLooG-0.16.3 32 Bit Libraries	149
10.19. CLooG-0.16.3 64 Bit	150
10.20. Zlib-1.2.7 32 Bit Libraries	151
10.21. Zlib-1.2.7 64 Bit	152

10.22. Binutils-2.23	153
10.23. GCC-4.6.3	156
10.24. Création d'un programme enveloppe multi-architecture (« Multiarch Wrapper »)	158
10.25. Sed-4.2.1	161
10.26. Bibliothèques Ncurses-5.9 32 bits	162
10.27. Ncurses-5.9 64 bits	164
10.28. Pkg-config-lite-0.27.1-1	167
10.29. Util-linux-2.22.1 32 bits	168
10.30. Util-linux-2.22.1 64 bits	169
10.31. Bibliothèques Procps-3.2.8 32 bits	173
10.32. Procps-3.2.8 64 bits	174
10.33. Bibliothèques E2fsprogs-1.42.6 32 bits	176
10.34. E2fsprogs-1.42.6 64 bits	177
10.35. Shadow-4.1.5.1	180
10.36. Coreutils-8.20	183
10.37. Iana-Etc-2.30	188
10.38. M4-1.4.16	189
10.39. Bibliothèques Bison-2.6.4 32 bits	190
10.40. Bison-2.6.4 64 bits	191
10.41. Libtool-2.4.2 32 bit Libraries	192
10.42. Libtool-2.4.2 64 bits	193
10.43. Bibliothèques Flex-2.5.37 32 bits	194
10.44. Flex-2.5.37 64 bits	195
10.45. IPRoute2-3.4.0	196
10.46. Bibliothèques Perl-5.16.2 32 bits	198
10.47. Perl-5.16.2 64 bits	200
10.48. Bibliothèques Readline-6.2 32 bits	203
10.49. Readline-6.2 64 bits	204
10.50. Autoconf-2.69	205
10.51. Automake-1.12.4	206
10.52. Bash-4.2	208
10.53. Bibliothèques Bzip2-1.0.6 32 bits	210
10.54. Bzip2-1.0.6 64 bits	211
10.55. Diffutils-3.2	213
10.56. Bibliothèques File-5.11 32 bits	214
10.57. File-5.11 64 bits	215
10.58. Gawk-4.0.1	216
10.59. Findutils-4.4.2	217
10.60. Bibliothèques Gettext-0.18.1.1 32 bits	218
10.61. Gettext-0.18.1.1 64 bits	219
10.62. Grep-2.14	221
10.63. Groff-1.21	222
10.64. Less-451	225
10.65. Gzip-1.5	226
10.66. IPutils-s20101006	227
10.67. Kbd-1.15.3	228
10.68. Make-3.82	230
10.69. Outils XZ Utils-5.0.4 32 bits	231
10.70. XZ Utils-5.0.4 64 bits	232
10.71. Man-1.6g	234

10.72. Kmod-10 32 Bit Libraries	236
10.73. Kmod-10 64 Bit	237
10.74. Patch-2.7.1	239
10.75. Psmisc-22.20	240
10.76. Libestr-0.1.0 32 Bit Libraries	241
10.77. Libestr-0.1.0 64 Bit	242
10.78. Libee-0.4.1 32 Bit Libraries	243
10.79. Libee-0.4.1 64 Bit	244
10.80. Rsyslog-6.2.2	245
10.81. Sysvinit-2.88dsf	248
10.82. Tar-1.26	251
10.83. Texinfo-4.13a	252
10.84. Bibliothèques Udev-182 32 bits	254
10.85. Udev-182 64 bits	255
10.86. Vim-7.3	257
10.87. GRUB-2.00	260
10.88. À propos des symboles de débogage	263
10.89. Supprimer les symboles de débogage	263
11. Initialiser les scripts de démarrage du système	264
11.1. Introduction	264
11.2. Scripts de démarrage pour CLFS 2.0-pre2	265
11.3. Comment fonctionnent ces scripts de démarrage ?	267
11.4. Configurer le script setclock	268
11.5. Configurer la console Linux	268
11.6. Gestion des périphériques et modules sur un système CLFS	269
11.7. Création de liens symboliques personnalisés vers les périphériques	272
11.8. Fichiers de démarrage du shell Bash	274
11.9. Setting Up Locale Information	274
11.10. Créer le fichier /etc/inputrc	276
12. Configuration du réseau	278
12.1. Configurer le script localnet	278
12.2. Personnaliser le fichier /etc/hosts	278
12.3. Création du fichier /etc/resolv.conf	279
12.4. Réseau DHCP ou Statique ?	279
12.5. Configuration d'un réseau statique	280
12.6. DHCPD-5.5.6	281
12.7. Configuration d'un réseau DHCP	282
13. Rendre le système CLFS amorçable	283
13.1. Introduction	283
13.2. Créer le fichier /etc/fstab	283
13.3. Linux-3.4.17	284
13.4. Rendre le système CLFS amorçable	286
14. La fin	287
14.1. La fin	287
14.2. Client de téléchargement	287
14.3. Redémarrer le système	288
14.4. Et maintenant ?	289
VI. Annexes	291
A. Acronymes et termes	292
B. Dépendances	295

C. x86 Dependencies	306
D. Raison de la présence des paquets	307
E. Open Publication License	312
Index	314

Préface

Avant-propos

Le projet Linux From Scratch a connu de nombreux changements ces dernières années. J'ai personnellement été impliqué dans le projet en 1999, période des versions 2.x. A cette époque, la procédure de construction consistait en la création de binaires statiques avec le système hôte, puis se chrooter et construire les binaires finaux sur la base de ceux statiques.

Plus tard on a commencé à utiliser le répertoire /static qui contenait les constructions statiques initiales, les séparant du système final, puis la procédure PureLFS développée par Ryan Oliver et Greg Schafer, introduisant une nouvelle procédure de construction de la chaîne d'outils qui sépare même nos constructions initiales de l'hôte. Enfin, LFS 6 a adopté le noyau Linux 2.6, la structure des périphériques dynamiques Udev, nettoyé les en-têtes du noyau et d'autres améliorations pour le système Linux From Scratch.

Le seul "défaut" dans LFS est qu'il a toujours été fondé sur un processeur de classe x86. Avec l'arrivée des processeurs Athlon 64 et Intel EM64T, la LFS constructible uniquement en x86 n'est plus idéale. Pendant ce temps, Ryan Oliver a développé et documenté une procédure par laquelle vous pourriez construire Linux pour n'importe quel système et à partir de n'importe quel système, en utilisant les techniques de la compilation croisée. Ainsi, le *Cross-Compiled LFS* (CLFS ou LFS croisé) est né.

CLFS suit les mêmes principes directeurs que le projet LFS a toujours suivis, comme celui selon lequel vous connaissez votre système à l'intérieur et à l'extérieur grâce au fait que vous ayez compilé votre système vous-même. En plus, pendant une construction CLFS, vous apprendrez des techniques avancées comme la construction croisée d'ensembles d'outils, le support multilib (bibliothèques 32 & 64 bits séparées), des architectures alternatives telles que Sparc, MIPS et Alpha et beaucoup plus.

Nous espérons que vous apprécierez la construction de votre propre système CLFS et les avantages résultant d'un système sur mesure selon vos besoins.

--

Jeremy Utey, gestionnaire de la version 1.x de CLFS (auteur de la Page)

Jonathan Norman, Gestionnaire des publications

Jim Gifford, Co-leader du projet CLFS

Ryan Oliver, Co-leader du projet CLFS

Joe Ciccone, Co-leader du projet CLFS

Jonathan Norman, Justin Knierim, Chris Staub, Matt Darcy, Ken Moffat,

Manuel Canales Esparcia, Nathan Coulson et William Harrington - Développeurs CLFS

Public visé

Il y a beaucoup de raisons qui pousseraient quelqu'un à vouloir lire ce livre. La raison principale est d'installer un système Linux à partir du code source. La question que beaucoup de personnes se posent est « pourquoi se fatiguer à installer manuellement un système Linux à partir de rien alors qu'il suffit de télécharger une distribution existante ? ». C'est une bonne question et c'est l'origine de cette section du livre.

Une raison importante de l'existence de CLFS est d'apprendre comment fonctionne un système Linux. Construire un système CLFS vous apprend tout ce qui fait que Linux fonctionne, et comment les choses interagissent et dépendent les unes des autres et, le plus important, vous apprend à le personnaliser afin qu'il soit à votre goût et réponde à vos besoins.

Un avantage clé de CLFS est qu'il permet aux utilisateurs d'avoir plus de contrôle sur leur système sans avoir à dépendre d'une implémentation créée par quelqu'un d'autre. Avec CLFS, *vous* êtes maintenant au volant et *vous* êtes capable de décider chaque aspect du système comme la disposition des répertoires ou la configuration des scripts de démarrage. Vous saurez également exactement où, pourquoi et comment les programmes sont installés.

Un autre avantage de CLFS est la possibilité de créer un système Linux très compact. Lors de l'installation d'une distribution habituelle, l'utilisateur est amené à inclure beaucoup de programmes qui ne seront peut-être jamais utilisés. Ces programmes occupent de l'espace disque et font parfois perdre de précieux cycles de processeur. Il n'est pas difficile de construire un système CLFS de moins de 100 Mo, ce qui est très petit comparé à la majorité des installations existantes. Cela vous semble-t-il toujours beaucoup ? Certains d'entre nous ont travaillé afin de créer un minuscule système CLFS. Nous avons installé un système spécialisé pour faire fonctionner le serveur web Apache ; l'espace disque total occupé était approximativement de 8 Mo voire moins. Avec plus de dépouillement encore, cela peut être ramené à 5 Mo ou moins. Essayez donc d'en faire autant avec une distribution courante ! C'est un des points bénéfiques de la conception de votre propre implémentation d'un système Linux.

Si nous devons comparer une distribution Linux à un hamburger que vous achetez à un restaurant fast-food, vous n'avez aucune idée de ce que vous mangez. CLFS ne vous donne pas un hamburger, mais la recette pour faire un hamburger. Cela permet aux utilisateurs d'inspecter la recette, d'enlever les ingrédients non désirés et, par la même occasion, de rajouter des ingrédients qui améliorent la saveur de ce hamburger. Quand vous êtes satisfait de la recette, vous passez à l'étape suivante en les combinant ensemble. Vous avez désormais la chance de pouvoir le faire de la façon dont vous le souhaitez : grillez-le, faites-le cuire au four, faites-le frire, faites-le au barbecue ou mangez-le cru.

Une autre analogie que nous pouvons utiliser est de comparer CLFS à une maison construite. CLFS fournit les plans de la maison, mais c'est à vous de la construire. CLFS vous donne la liberté d'ajuster les plans pendant tout le processus, le personnalisant suivant les besoins et préférences des utilisateurs.

Un autre avantage d'un système Linux personnalisé est un surcroît de sécurité. Vous compilerez le système complet à partir de la base, ce qui vous permet de tout vérifier, si vous le voulez, et d'appliquer tous les correctifs de sécurité désirés. Il n'est plus nécessaire d'attendre que quelqu'un d'autre vous fournisse un paquet réparant une faille de sécurité. À moins que vous examiniez vous-mêmes le correctif et que vous l'appliquiez, vous n'avez aucune garantie que le nouveau paquet ait été compilé correctement et résolve effectivement le problème.

Le but de Cross Linux From Scratch est de construire un système complet et utilisable, en ce qui concerne les fondations. Les lecteurs qui ne souhaitent pas construire leur propre système à partir de rien pourraient ne pas bénéficier des informations contenues dans ce livre. Si vous voulez seulement savoir ce qui se passe pendant le démarrage de l'ordinateur, nous vous recommandons le guide pratique « De la mise sous tension à l'invite de commande de Bash », disponible sur <http://www.traduc.org/docs/HOWTO/lecture/From-PowerUp-To-Bash-Prompt-HOWTO.html> ou, en anglais, <http://axiom.anu.edu.au/~okeefe/p2b/> ou sur le site du projet de documentation Linux (TLDP) à <http://www.tldp.org/HOWTO/From-PowerUp-To-Bash-Prompt-HOWTO.html>. Ce guide pratique construit un système qui est similaire à celui de ce livre mais qui se concentre strictement sur la création d'un système capable de démarrer jusqu'à l'invite de BASH. Prenez en compte vos objectifs. Si vous souhaitez construire un système Linux tout en apprenant, alors ce livre est votre meilleur choix possible.

Il existe trop de bonnes raisons de construire votre système CLFS pour pouvoir toutes les lister ici. Cette section n'aborde que la partie visible de l'iceberg. En continuant dans votre expérience de CLFS, vous trouverez la puissance réelle que donnent l'information et la connaissance.

Prérequis

Construire un système CLFS 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, le lecteur devrait 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 le lecteur dispose d'une connaissance raisonnable de l'utilisation et de l'installation de logiciels Linux. Une connaissance de base des architectures qui seront utilisées dans la procédure CLFS croisé et des systèmes d'exploitation hôtes utilisés est également nécessaire.

Comme le livre CLFS attend *au moins* ce simple niveau de connaissance, les différents forums de support CLFS 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 CLFS, 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.

- The Linux Users' Guide <http://www.linuxhq.com/guides/LUG/guide.html>

Ce guide couvre l'utilisation de différents logiciels Linux.

- The Essential Pre-Reading Hint http://hints.cross-lfs.org/index.php/Essential_Prereading

C'est une astuce écrite spécifiquement pour les nouveaux utilisateurs Linux. C'est principalement une liste de liens de sources excellentes 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

Vous devriez pouvoir construire un système CLFS à partir de presque tout système d'exploitation de type Unix. Votre système hôte devrait avoir les logiciels suivants avec la version minimum indiquée. Remarquez aussi que beaucoup de distributions mettront les en-têtes des logiciels dans des paquets séparés, ayant souvent la forme « [package-name]-devel » ou « [package-name]-dev ». Assurez-vous de les installer si votre distribution les fournit.

- **Bash-2.05a**
- **Binutils-2.12** (Les versions supérieures à 2.23 ne sont pas recommandées car elles n'ont pas été testées)
- **Bison-1.875**
- **Bzip2-1.0.2**
- **Coreutils-5.0**
- **Diffutils-2.8**
- **Findutils-4.1.20**
- **Gawk-3.1.5**
- **GCC 4.1** (Les versions supérieures à 4.6.3 ne sont pas recommandées car elles n'ont pas été testées)
- **Glibc-2.2.5** (Les versions supérieures à 2.15 ne sont pas recommandées car elles n'ont pas été testées)
- **Grep-2.5**
- **Gzip-1.2.4**
- **Linux 2.6.32** (construit avec GCC 4.1.2 ou supérieur)
- **Make-3.80**
- **Ncurses-5.3**
- **Patch-2.5.4**
- **Sed-3.0.2**
- **Tar-1.22**
- **Texinfo-4.4**
- **XZ-Uutils-4.999.8beta**

Pour voir si votre système hôte fournit des versions appropriées, créez et lancez le script suivant. Lisez attentivement la sortie pour voir les erreurs et vous assurer d'installer tous les paquets indiqués comme non trouvés :

```
cat > version-check.sh << "EOF"
#!/bin/bash

# Simple script to list version numbers of critical development tools

bash --version | head -n1 | cut -d" " -f2-4
echo -n "Binutils: "; ld --version | head -n1 | cut -d" " -f3-
bison --version | head -n1
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
gcc --version | head -n1
ldd $(which ${SHELL}) | grep libc.so | cut -d ' ' -f 3 | ${SHELL} | head -n 1
grep --version | head -n1
gzip --version | head -n1
uname -s -r
make --version | head -n1
tic -V
patch --version | head -n1
sed --version | head -n1
tar --version | head -n1
makeinfo --version | head -n1
xz --version | head -n1
echo 'main(){}' | gcc -v -o /dev/null -x c - > dummy.log 2>&1
if ! grep -q 'error' dummy.log; then
    echo "Compilation réussie" && rm dummy.log
else
    echo 1>&2 "Compilation ÉCHOUÉE - installez peut-être plus de paquets de développement
    Si vous voulez vous pouvez lire le journal dummy pour plus de détails."
fi
EOF

bash version-check.sh 2>errors.log &&
[ -s errors.log ] && echo -e "\nLes paquets suivants sont introuvables :\n$(cat errors.log)"
```

Typographie

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

```
./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.

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

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

Emphasis

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

<http://cross-lfs.org/>

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

```
cat > $CLFS/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 `$CLFS/etc/group` à partir de ce qui est saisi jusqu'à ce que la séquence de fin de fichier (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.

`passwd(5)`

Ce format est utilisé pour faire référence à une page de manuel spécifique (noté après comme une page « man »). Le nombre entre parenthèses indique une section spécifique à l'intérieur de **man**. Par exemple, **passwd** a deux pages man. Pour les instructions d'installation de LFS, ces deux pages man seront situées dans `/usr/share/man/man1/passwd.1` et `/usr/share/man/man5/passwd.5`. Ces deux pages man comprennent des informations différentes. 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 man n'ont pas de noms de page dupliqués dans les différentes sections. Du coup, **man [nom du programme]** est généralement suffisant.

Structure

Ce livre est divisé selon les parties suivantes.

Partie I - Introduction

La première partie donne quelques informations importantes, comme par exemple sur la façon d'installer Linux From Scratch Croisé. 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 et téléchargement des paquets.

Partie III - Fabrication des outils de compilation croisée

La troisième partie vous montre comment régler une chaîne d'outils de compilation croisée. Ces outils peuvent s'exécuter sur votre système hôte mais vous permettent de construire des paquets qui s'exécuteront sur votre système cible.

Partie IV - Construction des outils de base

La quatrième partie explique comment construire une chaîne d'outils amenés à fonctionner sur votre système cible. Ce sont les outils qui vous permettront de construire un système opérationnel sur votre système cible.

Partie V - Construction du système CLFS

La cinquième partie guide le lecteur à travers la construction du système CLFS—la compilation et l'installation de tous les paquets un par un, l'initialisation des scripts de démarrage et l'installation du noyau. Le système Linux qui en résulte est une base sur laquelle d'autres logiciels peuvent être construits pour étendre le système comme désiré. À la fin de ce livre, il y a une liste de références facile à utiliser de tous les programmes, bibliothèques et fichiers importants qui ont été installés, .

Annexes

Les annexes contiennent des informations qui ne vont nulle part ailleurs dans le livre. L'annexe A contient les définitions d'acronymes et de termes utilisés dans le livre—les annexes B et C ont des informations sur les dépendances des paquets et l'ordre de construction. Il se peut que certaines architectures aient des annexes supplémentaires pour des questions qui leur sont propres.

Errata

Le logiciel utilisé pour créer un système CLFS est constamment mis à jour et amélioré. Les avertissements pour la sécurité et les corrections de bogues pourraient survenir après la sortie du livre CLFS. Pour vérifier si les versions des paquets ou les instructions de cette version de CLFS ont besoin de modifications pour corriger les vulnérabilités en terme de sécurité ou toute autre correction de bogue, merci de visiter <http://trac.cross-lfs.org/wiki/errata> 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 CLFS.

Partie I. Introduction

Chapitre 1. Introduction

1.1. Remerciements LFS Croisé

L'équipe CLFS aimerait remercier les gens qui nous ont aidé à faire de ce livre ce qu'il est aujourd'hui.

Nos Leaders :

- Ryan Oliver - Développeur de la procédure de construction.
- Jim Gifford - Développeur leader.
- Joe Ciccone - Développeur leader
- Jeremy Utley - Responsable de publication de la série 1.x.

Notre équipe CLFS :

- Nathan Coulson - scripts de démarrage.
- Matt Darcy - constructions de x86, X86_64 et Sparc.
- Manuel Canales Esparcia - livre XML.
- Karen McGuiness - Relecteur.
- Jonathan Norman - x86, x86_64, PowerPC & UltraSPARC.
- Jeremy Huntwork - constructions de PowerPC, x86, Sparc.
- Justin Knierim - architecte du site Internet.
- Ken Moffat - constructions de PowerPC and X86_64. Développeur de l'astuce Pure 64.
- Alexander E. Patrakov - intégration d'Udev/Hotplug
- Chris Staub - constructions de x86. Leader du contrôle qualité (*Quality Control*).
- Zack Winkles - Travail sur le livre instable.
- William Harrington - construit x86, x86_64, PowerPC, Sparc, Mips.

À l'extérieur de l'équipe de développement

- Jürg Billeter - Test et aide pour le développement du paquet Linux Headers
- Richard Downing - Test, correction de fautes de frappe et de contenu.
- Peter Ennis - Corrections de fautes de frappe et de contenu.
- Tony Morgan - Corrections de fautes de frappe et de contenu.

L'équipe CLFS aimerait aussi remercier les contributions de gens issus de clfs-dev@lists.cross-lfs.org et des listes de diffusion associées qui ont fourni des corrections techniques et éditoriales appréciables lors du test du livre LFS croisé.

- G. Moko - Mise à jour du texte et correction des fautes de frappe
- Maxim Osipov - Test de MIPS.
- Doug Ronne - Diverses corrections de x86_64.
- William Zhou - Mise à jour du texte et correction des fautes de frappe
- Theo Schneider - Test du paquet Linux Headers

Merci à tous pour votre soutien.

1.2. Comment construire un système CLFS

Le système CLFS sera construit en utilisant une distribution Linux déjà installée (telle que Debian, Fedora, Mandrake, Red Hat, SuSE ou Ubuntu). 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 un LiveCD. La plupart des distributions offrent un LiveCD, qui fournissent un environnement sur lequel vous pouvez ajouter les outils nécessaires, ce qui vous permet de suivre sans problèmes les instructions de ce livre. Souvenez-vous que si vous redémarrez le liveCD, vous devrez reconfigurer l'environnement hôte avant de continuer votre construction.

Le *Preparing a New Partition* de ce livre décrit comment créer une nouvelle partition native Linux et un système de fichiers, c'est-à-dire un emplacement où le nouveau système CLFS sera compilé et installé. Le *Packages and Patches* explique quels paquets et correctifs ont besoin d'être téléchargés pour construire un système CLFS et comment les stocker sur le nouveau système de fichiers. *Final Preparations* traite de l'initialisation d'un environnement de travail approprié. Merci de lire le *Final Preparations* attentivement car il explique plusieurs points importants qu'un développeur doit savoir avant de commencer à travailler sur le *Constructing Cross-Compile Tools* et au-delà.

Constructing Cross-Compile Tools explique l'installation des outils de compilation croisée qui seront construits sur l'hôte mais qui pourront compiler des programmes qui se lancent sur la machine cible. Ces outils de compilation croisée seront utilisés pour créer un système temporaire et minimal, qui sera la base de la construction du système CLFS final. Certains de ces paquets sont nécessaires pour résoudre des dépendances circulaires—par exemple, pour compiler un compilateur, vous avez besoin d'un compilateur.

La procédure de construction des outils de compilation croisée implique tout d'abord de construire et d'installer tous les outils nécessaires pour créer un système de construction pour la machine cible. Avec ces outils de compilation croisée, nous éliminons toute dépendance de la chaîne d'outils par rapport à notre distribution hôte.

Après avoir construit nos « outils croisés », nous commençons à construire un système opérationnel très minimal dans `/tools`. Ce système minimal sera construit en utilisant la chaîne d'outils croisés dans `/cross-tools`.

Dans le *Installing Basic System Software*, on construit le système CLFS complet. Selon le système pour lequel vous faites une compilation croisée, soit vous démarrerez le système temporaire minimal sur la machine cible, soit vous vous chrootez dedans.

Le programme **chroot** (*change root*, changer de racine) est utilisé pour rentrer dans un environnement virtuel et démarrer un nouveau shell dont le répertoire racine sera initialisé à la partition CLFS. Cela ressemble beaucoup à un démarrage et à une demande au noyau de monter la partition CLFS en tant que partition racine. Le principal avantage est que « chrooter » permet à celui qui construit de continuer à utiliser l'hôte pendant que CLFS se construit. Tout en attendant que la compilation d'un paquet se termine, un utilisateur peut ouvrir sur une autre console virtuelle (VC) ou un bureau X et continuer à utiliser l'ordinateur normalement.

Certains systèmes ne peuvent être compilés en se chrootant et doivent donc être démarrés. Généralement, si vous construisez pour une architecture différente du système hôte, vous devez redémarrer car le noyau ne supportera probablement pas la machine cible. Un redémarrage oblige à installer quelques paquets supplémentaires nécessaires pour un démarrage, à installer des scripts de démarrage et à construire un noyau minimal. Nous décrivons aussi des méthodes de démarrage alternatives dans la Section 7.19, « Que faire ensuite »

Pour finir l'installation, on initialise les scripts de démarrage CLFS dans le *Setting Up System Bootscripts*, le noyau ainsi que le chargeur de démarrage dans le *Making the CLFS System Bootable*. Le *The End* contient des informations pour aller au-delà de l'expérience CLFS, plus loin que le livre. Après avoir effectué cette étape du livre, l'ordinateur sera prêt à redémarrer dans le nouveau système CLFS.

C'est en gros la procédure. Des informations détaillées sur chaque étape sont données dans les chapitres suivants, ainsi que les descriptions des paquets. Les points qui paraissent complexes seront clarifiés et tout prendra du sens au fur et à mesure que le lecteur se lance dans l'aventure CLFS.

1.3. Historique des gros changements

Ceci est la version GIT-20130401 du livre *Cross-Compiled Linux From Scratch* (Linux From Scratch Croisé), datant du avril 01, 2013. Si ce livre a plus de six mois, une version plus récente et meilleure est probablement disponible. Pour la trouver, merci de vérifier un des miroirs sur <http://trac.cross-lfs.org/>.

Ci-dessous une liste des changements détaillés effectués depuis la version précédente du livre.

Liste des modifications :

- 2 mars 2013
 - [William Harrington] - Mise à jour de l'avant-propos.
 - [William Harrington] - Mise à jour de l'emplacement de l'errata.
- 16 février 2013
 - [William Harrington] - Suppression d'une entrée inutile de config.cache aux sections Démarrage et chroot pour util-linux.
- 13 février 2013
 - [William Harrington] - Mise à jour de l'emplacement du téléchargement de dhcpcd.
- 9 février 2013
 - [William Harrington] - Ajout des commandes des suites de tests à udev du système final.
 - [William Harrington] - Mise à jour de la section sur la remarque d'iana-etc pour le correctif get.
- 8 février 2013
 - [William Harrington] - Déplacement de gawk avant findutils dans le système final pour la suite de tests coverbde de findutils.
 - [William Harrington] - Déplacement de less avant gzip dans le système final pour la suite de tests coverage de gzip.
 - [William Harrington] - Mise à jour de l'entrée de la suite de tests pour rsyslog dans le système final.
- 6 février 2013
 - [William Harrington] - Modification des informations de la suite de tests pour ncurses du système final.
 - [William Harrington] - Modification des informations de la suite de tests pour util-linux du système final.
 - [William Harrington] - Modification des informations de la suite de tests pour coreutils du système final.
- 3 février 2013
 - [William Harrington] - Modification de la locale du pays en la locale du territoire. Le pays n'est plus valide.
- 27 janvier 2013
 - [William Harrington] - Ajout d'une nouvelle ligne au fstab de la méthode du redémarrage et à la section Démarrage.
 - [William Harrington] - Correction d'un lien ../run -> /var/run incorrect.
- 27 décembre 2012
 - [William Harrington] - Déplacement de ProcPS avant E2fsprogs car la suite de tests exige ps.

- 13 décembre 2012
 - [Chris] - Suppression d'un paramètre --enable-add-ons redondant dans l'installation d'EGLIBC.
- 18 novembre 2012
 - [Chris] - Nombreuses mises à jour de listes des programmes installés
- 17 novembre 2012
 - [William Harrington] - Passage du kill pendant l'installation de Procps au système final.
 - [William Harrington] - Suppression de sulogin, de mountpoint, d'utmpdump et de wall de sysvinit.
- 12 novembre 2012
 - [Chris] - Suppression de --disable-perl-regexp inutile dans le grep du système temporaire.
- 5 novembre 2012
 - [William Harrington] - Mise à jour du correctif de la branche Mise à jour de gcc vers r193147.
 - [William Harrington] - Mise à jour de binutils vers 2.23.
 - [William Harrington] - Suppression du correctif de la branche Mise à jour de Binutils 2.22.
 - [William Harrington] - Modification de la construction de coreutils du système temporaire.
- 4 novembre 2012
 - [William Harrington] - Mise à jour du correctif de la branche Mise à jour de bash au niveau 39.
- 2 novembre 2012
 - [William Harrington] - Désactivation des programmes login et su d'util-linux.
 - [William Harrington] - Modification du sed de hwclock pour util-linux.
 - [William Harrington] - Modification de la suite de tests de Coreutils.
- 1 novembre 2012
 - [William Harrington] - Mise à jour de Patch vers 2.7.1.
 - [William Harrington] - Mise à jour de Perl vers 5.16.2.
 - [William Harrington] - Mise à jour de Pkg-Config-Lite vers 0.27.1-1.
 - [William Harrington] - Mise à jour de Psmisc vers 22.20.
 - [William Harrington] - Mise à jour du correctif de la branche Mise à jour de Readline au niveau 004.
 - [William Harrington] - Mise à jour d-Util-linux vers 2.22.1.
 - [William Harrington] - Mise à jour de Vim-7.3 au niveau de correctif 712.
 - [William Harrington] - Noyau Linux vers 3.4.17.
 - [William Harrington] - Suppression du correctif de test.
- 31 octobre 2012
 - [William Harrington] - Mise à jour d'eglibc à la révision 21435.
 - [William Harrington] - Mise à jour d'automake vers 1.12.4.
 - [William Harrington] - Mise à jour de bison vers 2.6.4.
 - [William Harrington] - Mise à jour de coreutils vers 8.20.
 - [William Harrington] - Mise à jour d'e2fsprogs vers 1.42.6.
 - [William Harrington] - Mise à jour de gzip vers 1.5.
 - [William Harrington] - Mise à jour de kmod vers 510.

- [William Harrington - Mise à jour de less vers 451.
- [William Harrington - Mise à jour de man-pages vers 3.43.
- [William Harrington - Mise à jour de mpc vers 1.0.1.
- 25 octobre 2012
 - [Chris] - Mise à jour de "Et maintenant ?" pour refléter le changement de nom de freshmeat.net.
- 23 octobre 2012
 - [William Harrington} - Ajout d'un correctif pour M4 dans les outils croisés.
- 17 octobre 2012
 - [William Harrington} - Modification de la commande de suite de tests de coreutils pour su.
- 15 octobre 2012
 - [William Harrington} - Déplacement de shadow avant coreutils
- 17 septembre 2012
 - [William Harrington] - Modification de la référence à ncftp dans la page Clients de téléchargement pour lier à la page cftp de cblfs.
 - [William Harrington] - Mise à jour de la version du noyau Linux de Update linux kernel version from 3.4.9 to 3.4.11.
- 14 septembre 2012
 - [William Harrington] - Mise à jour de la somme de contrôle et de la taille d'iproute 3.4.0 libdir.
 - [William Harrington] - Mise à jour de la liste de liens de téléchargement et ajustement du texte d'introduction des paquets et des correctifs.
- 11 septembre 2012
 - [William Harrington] - Installation des en-têtes liées à NIS et RPC dans l'installation d'eglibc/eglibc-64bit dans les outils croisés et le système final.
- 7 septembre 2012
 - [William Harrington] - Suppression de --with-rootlibdir du configure de kmod au Ch 7.
 - [William Harrington] - Désactivation de la construction des bibliothèques statiques quand il le faut au moment de la compilation des outils croisés.
 - [William Harrington] - Suppression de la création des liens vers passwd et login dans Shadow, section Démarrage du Ch7.
 - [William Harrington] - Ajout de passwd à la chaîne des liens symboliques créés dans la section Démarrage du Ch7.
- 6 septembre 2012
 - [William Harrington] - Correction de la commande /var/run /run à la partie Création de fichiers dans la section Si vous allez redémarrer.
 - [William Harrington] - Ajout de shadow à la section Si vous allez redémarrer.
 - [William Harrington] - Suppression d'enable-login-utils d'util-linux à la section Si vous allez redémarrer.
- 4 septembre 2012
 - [William Harrington] - Ajout d'une commande à Bison du système final pour ajouter une variable à config.cache.
 - [William Harrington] - Correction de la somme de contrôle MD5 de zlib 1.2.7.

- [William Harrington] - Mise à jour des options de configure d'Udev aux sections Démarrage et Système final.
- [William Harrington] - Mise à jour des scripts de démarrage pour les rendre adaptés aux mises à jour d'udev.
- 3 septembre 2012
 - [William Harrington] - Ajout d'une page sur le nouveau Client de téléchargement dans la section La fin.
- 30 août 2012
 - [William Harrington] - Passage des prérequis du système hôte du livre, concernant le noyau Linux, à la version 2.6.32 ou supérieur.
- 29 août 2012
 - [William Harrington] - Modification de la ligne de configuration de PPL lors de la compilation croisée.
 - [William Harrington] - Passage des prérequis du système hôte du livre, concernant le noyau Linux, à la version 2.6.32.
 - [William Harrington] - Mise à jour des instructions d'eglibc et du texte concernant eglibc pour le support du noyau 2.6.32.
 - [William Harrington] - Ajout de --with-default-terminfo-dir=/usr/share/terminfo à ncurses du système final du fait de modifications dans la branche Mise à jour.
- 28 août 2012
 - [William Harrington] - Modification du programme groups de shadow et désactivation de la section des man-pages.
- 27 août 2012
 - [William Harrington] - Création et ajout du correctif de la branche Mise à jour de binutils 2.22.
 - [William Harrington] - Mise à jour de la version de bison vers 2.6.2.
 - [William Harrington] - Mise à jour de la version de coreutils vers 8.19.
 - [William Harrington] - Mise à jour de la version d'elfsprogs vers 1.42.5.
 - [William Harrington] - Mise à jour de la version de flex vers 2.5.37.
 - [William Harrington] - Création et ajout au livre du correctif de la branche mise à jour de gcc 4.6.3.
 - [William Harrington] - Mise à jour de la version de grep vers 2.14.
 - [William Harrington] - Mise à jour de la version de grub vers 2.00.
 - [William Harrington] - Mise à jour de la version d'iproute2 vers 3.4.0 et régénération du correctif Iproute2.
 - [William Harrington] - Mise à jour de la version de kmod vers 9.
 - [William Harrington] - Mise à jour de la version de linux vers 3.4.9.
 - [William Harrington] - Mise à jour de la version de man-pages vers 3.42.
 - [William Harrington] - Mise à jour de la version de mpc vers 1.0.
 - [William Harrington] - Mise à jour de la version de mpfr vers 3.1.1.
 - [William Harrington] - Mise à jour du correctif de la branche mise à jour de ncurses.
 - [William Harrington] - Mise à jour de la version de perl vers 5.16.1 et régénération des correctifs libc et multilib.
 - [William Harrington] - Remplacement de pkg-config par pkg-config-lite 0.27-1.
 - [William Harrington] - Suppression du paquet glib du livre.

- [William Harrington] - Mise à jour de la version de PPL vers 0.12.1.
- [William Harrington] - Mise à jour de la version de psmisc vers 22.19.
- [William Harrington] - Mise à jour de la version de rsyslog vers 6.2.2.
- [William Harrington] - Mise à jour de la version de shadow vers 4.1.5.1.
- [William Harrington] - Mise à jour de la version d'util-linux vers 2.21.2.
- [William Harrington] - Mise à jour de la version d'xz vers 5.0.4.
- [William Harrington] - Modification de la ligne de commande de configuration de PPL.
- [William Harrington] - Suppression du correctif gcc44 de flex.
- [William Harrington] - Ajout de /run/shm aux sections de création des répertoires du livre.
- [William Harrington] - Suppression des variables GLIB CFLAGS et LIBS de la ligne de configuration de pkg-config.
- [William Harrington] - Suppression du sed pour les pages de manuel russes dans shadow.
- [William Harrington] - Remplacement de la méthode de chiffrement MD5 par celle SHA512 pour les définitions de connexion.
- [William Harrington] - Modification de la ligne de commande de configuration de shadow.
- [William Harrington] - Mise à jour de la ligne de commande de configuration d'udev pour une bonne installation.
- [William Harrington] - Mise à jour du PPL croisé et temporaire pour détecter et utiliser le bon gmp, par sécurité.
- [William Harrington] - Ajout d'une remarque sur l'installation d'iana-etc.
- [William Harrington] - Suppression d'une ligne YYENABLE_NLS inutile dans Bison.
- 26 août 2012
 - [William Harrington] - Correction de la construction d'IPutils pour que rdisc soit créé et suppression des entrées rdisc multiples pour les livres non multilib et ajout de rdisc aux livres multilib.
 - [William Harrington] - Ajout d'une remarque sur libee dans tous les livres et clarification de ce point avec plus d'explications.
- 22 août 2012
 - [William Harrington] - Abandon de la référence à bash dans le script de version hostreqs pour utiliser la variable \$SHELL.
- 18 août 2012
 - [William Harrington] - Passage d'automake à 1.12.3.
- 15 août 2012
 - [William Harrington] - Mise à jour de l'emplacement de la liste de téléchargements.
- 13 août 2012
 - [William Harrington] - Ajout de la compression xz et zlib au kmod de la méthode du redémarrage.
- 11 août 2012
 - [William Harrington] - Modification du bloc de commandes de configure de la section d'Udev dans la méthode du redémarrage, afin que le copier-coller fonctionne bien.
 - [William Harrington] - Modification de la commande configure de la section kmod de la méthode du redémarrage pour installer libkmod dans /tools/lib et non dans /lib.
- 06 août 2012

- [William Harrington] - Ajustement de la commande d'installation de XZ dans le système final pour installer correctement le fichier pkgconfig lzma au bon endroit.
- [William Harrington] - Mise à jour du script de vérification des versions pour qu'il trouve la version de libc avec les hôtes qui utilisent d'autres chemins que /lib et /lib64, tels que les distribs multiarchitectures.
- 2 août 2012
 - [William Harrington] - Ajout de \${CLFS} à la commande ln -s /run /var/run dans la méthode de redémarrage.
- 31 juillet 2012
 - [William Harrington] - Ajout de moi-même à la page des remerciements.
- 23 juillet 2012
 - [William Harrington] - Ajout de xz-utils aux exigences du système hôte.
- 21 juillet 2012
 - [William Harrington] - Mise à jour du correctif de vim 7.3 vers le niveau 608.
 - [William Harrington] - Mise à jour du correctif de 4.2 vers le niveau 37.
 - [William Harrington] - Modification de la description de chatr de e2fsprogs.
 - [William Harrington] - Suppression des correctifs d'eglibc-2.15-r17386-dl_dep_fix-1.patch, inutile.
 - [William Harrington] - Suppression des options de configuration graphite de Binutils.
 - [William Harrington] - Mise à jour d'automake vers la version 1.12.2.
- 18 juillet 2012
 - [Jonathan] - Suppression dans la liste de correctifs d'une branche mise à jour de GCC qui n'existe pas.
- 10 juin 2012
 - [Jonathan] - Ajout d'un correctif pour mettre à jour les fichiers Protocol et Service par défaut pour Iana-etc.
 - [Jonathan] - Ajout de devtmpfs et de firmware_install au noyau.
 - [Jonathan] - Mise à jour de la configuration finale de Coreutils pour permettre de le construire en tant qu'utilisateur root.
- 6 juin 2012
 - [Jonathan] - Mise à jour de Coreutils de la 8.15 à la 8.16.
 - [Jonathan] - Mise à jour d'Util-linux de la 2.20 à la 2.20.1.
- 4 juin 2012
 - [Jonathan] - Mise à jour de DHCPD de la 5.5.4 à la 5.5.6.
 - [Jonathan] - Mise à jour d'Udev de la 181 à la 182.
 - [Jonathan] - Mise à jour de Libee de la 0.3.2 à la 0.4.1.
 - [Jonathan] - Mise à jour de PSMisc de la 22.15 à la 22.17.
 - [Jonathan] - Mise à jour de Kmod de la 6 [la 8.
 - [Jonathan] - Mise à jour d'Automake de la 1.11.3 à la 1.12.1.
 - [Jonathan] - Mise à jour d'Autoconf de la 2.68 à la 2.69.
 - [Jonathan] - Mise à jour d'IPRoute2 de la 3.2.0 à la 3.3.0.
 - [Jonathan] - Mise à jour d'E2fsprogs de la 1.41 à la 1.42.3.
 - [Jonathan] - Mise à jour de Glib2 de la 2.28.6 à la 2.28.8.

- [Jonathan] - Mise à jour de Man-Pages de la 3.35 à la 3.41.
- [Jonathan] - Mise à jour de Grep de la 2.10 à la 2.12.
- [Jonathan] - Mise à jour de Gawk de la 4.0.0 à la 4.0.1.
- [Jonathan] - Mise à jour de Zlib de la 1.2.6 à la 1.2.7.
- [Jonathan] - Mise à jour de GCC de la 4.6.2 à la 4.6.3.
- [Jonathan] - Mise à jour de GMP de la 0.11.2 à la 0.12.1.
- [Jonathan] - Mise à jour de File de la 5.10 à la 5.11.
- [Jonathan] - Mise à jour de Linux de la 3.2.6 à la 3.3.7.
- [Jonathan] - Mise à jour du correctif de la branche Mise à jour de Bash vers -4.
- [Jonathan] - Mise à jour du correctif de la branche Mise à jour de Vim vers -4.
- 15 avril 2012
 - [Jonathan] - Ajout de /run au livre.
 - [Jonathan] - Mise à jour d'Udev de 168 vers 181.
- 14 Mars 2012
 - [Jonathan] - Remplacement de Module-Init-tools par Kmod.
- 3 mars 2012
 - [Jonathan] - Mise à jour d'Eglibc 2.15 de la r16526 à la r17386 et correction du nom du répertoire.
- 29 février 2012
 - [Jonathan] - Ajout de Login aux liens créés dans le méthode avec redémarrage - merci Code Monkey.
 - [Jonathan] - Correction de problèmes avec la méthode du redémarrage et e2fsprogs de mutlib.
- 20 février 2012
 - [Jonathan] - Ajout de --without-nscd à Shadow.
 - [Jonathan] - Ajout de --with-ppl à Binutils en outils croisés et temp.
- 18 février 2012
 - [Jonathan] - Activation de la suite de tests de Patch.
 - [Jonathan] - Diffutils inclut maintenant une suite de tests.
 - [Jonathan] - Mise à jour de la branche Mise à jour de Readline vers -2.
 - [Jonathan] - Correction de problèmes de compilation avec IPRoute2 par la suppression d'en-têtes libnl inutiles.
- 17 février 2012
 - [Jonathan] - Ajout d'un correctif à Iana-etc et mise à jour de la commande.
 - [Jonathan] - Remplacement de ClooG-PPL par ClooG-0.16.3.
- 16 février 2012
 - [Jonathan] - Mise à jour de la branche Mise à jour de Ncurses vers -3.
 - [Jonathan] - Ajout d'un correctif à Eglibc pour corriger des problèmes de mémoire avec ALSA.
 - [Jonathan] - Mise à jour de Man-pages vers 3.35.
- 15 février 2012
 - Updated Correctif de la branche mise à jour de Vim à -3.
 - Mise à jour du correctif de la branche Mise à jour de Bash vers -3.

- Mise à jour d'Automake vers 1.11.3.
- Mise à jour de Binutils vers 2.22.
- Mise à jour de Coreutils vers 8.15.
- Mise à jour de DHCPD vers 5.5.4.
- Mise à jour de Diffutils vers 3.2.
- Mise à jour d'Eglibc vers 2.15.
- Mise à jour d'E2fsprogs vers 1.4.2.
- Mise à jour de File vers 5.10.
- Mise à jour de Gawk vers 4.0.
- Mise à jour de GCC vers 4.6.2.
- Mise à jour de GMP vers 5.0.4.
- Mise à jour de Grep vers 2.10.
- Mise à jour de Grep vers 2.10.
- Mise à jour de Iproute2 vers 3.2.0.
- Mise à jour de Less vers 444.
- Mise à jour de Libee vers 0.3.2.
- Mise à jour de Libtool vers 2.4.2.
- Mise à jour de Linux vers 3.2.x.
- Mise à jour de Module-init-tools vers 3.15.
- Mise à jour de MPFR vers 3.1.0.
- Mise à jour de Perl vers 5.14.2.
- Mise à jour de PSmisc vers 22.15.
- Mise à jour de Rsyslog vers 6.2.0.
- Mise à jour de Shadow vers 4.1.5.
- Mise à jour de TCL vers 8.5.11.
- Mise à jour d'Util-linux vers 2.20.
- Mise à jour d'XZ-Utils vers 5.0.3.
- Mise à jour de Zlib vers 1.2.6.
- 15 février 2012
 - [Jonathan] - Redémarrage de l'historique des changements, voir le livre 1.2.0 pour l'ancien historique.

1.4. Historique des modifications pour x86_64

La liste ci-dessous contient les changements spécifiques à cette architecture effectués depuis la dernière version du livre. Pour les changements généraux, voir Master Changelog,

Liste des modifications :

- 27 janvier 2013
 - [William Harrington] - Correction des instructions d'util-linux pour la méthode du redémarrage.
- 11 septembre 2012

- [William Harrington] - Suppression de la correction de l'entête dans pour eglibc utilib 32 bits dans le système final.
- 7 septembre 2012
 - [William Harrington] - Suppression de la remarque sur flex qui n'est pas installé lors de l'installation de Grub au ch7.
 - [William Harrington] - Correction des instructions d'installation de grub au Ch 7.
- 4 septembre 2012
 - [William Harrington] - Mise à jour des instructions pour Grub dans la section Démarrage.
- 26 août 2012
 - [William Harrington] - Suppression de commandes de création de liens du Kmod 32 bits, car ils sont créés dans la construction du Kmod 64 bits final.
- 7 août 2012
 - [William Harrington] - Modification de la commande configure dans xz du système final 32 bits.
- 31 juillet 2012
 - [William Harrington] - Suppression des binaires et du lien enveloppes multiarch.
- 21 juillet 2012
 - [William Harrington] - Ajout du correctif HZ à procps.
 - [William Harrington] - Suppression du déplacement du binaire logger d'util-linux 32 bits en multilib.
 - [William Harrington] - Suppression du lien de test testcase de l'enveloppe multiarch.
 - [William Harrington] - Correction des instructions de préparation d'udev 32 bits et 64 bits.
 - [William Harrington] - Correction des instructions de préparation de kmod.
 - [William Harrington] - Correction d'un problème de construction rpc dans la construction en 32 bits d'eglibc dans le système final.
- 15 février 2012
 - [Jonathan] - Historique des changements redémarré, voir le livre 1.2.0 pour l'ancien historique.

1.5. Ressources

1.5.1. FAQ

Si vous rencontrez des erreurs lors de la construction du système CLFS, si vous avez des questions ou si vous pensez qu'il y a une faute de frappe dans ce livre, merci de commencer par consulter la Foire aux Questions (FAQ) sur <http://trac.cross-lfs.org/wiki/faq>.

1.5.2. Listes de diffusion

Le serveur `cross-lfs.org` gère quelques listes de diffusion utilisées pour le développement du projet CLFS. Ces listes incluent, entre autres, les listes de développement et de support. Si la FAQ ne résout pas votre problème, vous pouvez chercher sur les listes CLFS sur les Archives Mail <http://www.mail-archive.com>. Vous pouvez trouver les listes de diffusion par le lien suivant :

<http://www.mail-archive.com/index.php?hunt=clfs>

Pour connaître les listes disponibles, les conditions d'abonnement, l'emplacement des archives et d'autres informations, allez sur <http://trac.cross-lfs.org/wiki/lists>.

1.5.3. Serveur de nouvelles

LFS croisé ne maintient pas son propre serveur de nouvelles mais nous fournissons un accès via gmane.org <http://gmane.org>. Si vous voulez vous abonner aux listes Cross-LFS par un lecteur de nouvelles, vous pouvez utiliser gmane.org. Vous pouvez trouver la recherche gmane pour CLFS avec le lien suivant :

<http://dir.gmane.org/search.php?match=clfs>

1.5.4. IRC

Plusieurs membres de la communauté CLFS offrent une assistance sur le réseau IRC (Internet Relay Chat) de notre communauté. Avant d'utiliser ce support, merci de vous assurer qu'on n'a pas répondu à votre question dans la FAQ CLFS ou dans les archives de la liste de diffusion. Vous pouvez trouver le réseau IRC sur chat.freenode.net. Le canal de support pour lfs croisé se nomme `#cross-lfs`. Si vous avez besoin de montrer aux gens la sortie de vos problèmes, merci d'utiliser <http://pastebin.cross-lfs.org> et de vous référer au lien pastebin lorsque vous posez vos questions.

1.5.5. Sites miroirs

Le projet CLFS a un bon nombre de miroirs configurés tout autour du monde pour faciliter l'accès au site web ainsi que le téléchargement des paquetages requis. Merci de visiter le site web de CLFS sur <http://trac.cross-lfs.org/wiki/mirrors> pour les miroirs de CLFS.

1.5.6. Contacts

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

1.6. Aide

Si vous rencontrez une erreur ou si vous vous posez une question en travaillant avec ce livre, vérifiez la FAQ sur <http://trac.cross-lfs.org/wiki/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://hints.cross-lfs.org/index.php/Errors>.

Nous avons aussi une formidable communauté CLFS, volontaire pour offrir une assistance via les listes de discussion et IRC (voir la section Section 1.5, « 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.

1.6.1. 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, GIT-20130401)
- La distribution hôte (et sa version) que vous utilisez pour créer CLFS
- L'architecture de l'hôte et de la cible.
- La valeur des variables d'environnement `$CLFS_HOST`, `$CLFS_TARGET`, `$BUILD32` et `$BUILD64`.
- Le paquet ou la section où le problème a été rencontré
- Le message d'erreur exact ou le symptôme reçu. Voir Section 1.6.3, « Problèmes de compilation » pour un exemple.

- Notez si vous avez dévié du livre. Un changement de version de paquet ou même un changement mineur sur une commande est considéré comme une déviation.



Remarque

Dévier du livre ne signifie *pas* que nous n'allons pas vous aider. Après tout, CLFS 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.

1.6.2. 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.

1.6.3. Problèmes de compilation

L'affichage à l'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 de **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/clfs/usr/share/locale:.\"
-DLOCALEDIR=\"/mnt/clfs/usr/share/locale\"
-DLIBDIR=\"/mnt/clfs/usr/lib\"
-DINCLUDEDIR=\"/mnt/clfs/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':
/clfs/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 `/clfs/tmp/make-3.79.1'
make[1]: *** [all-recursive] Error 1
make[1]: Leaving directory `/clfs/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://catb.org/~esr/faqs/smart-questions.html>. Lisez et suivez les astuces de ce document pour accroître vos chances d'obtenir l'aide dont vous avez besoin.

Partie II. Préparation de la construction

Chapitre 2. Préparez 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, CLFS est habituellement installé dans une partition dédiée. L'approche recommandée pour la construction d'un système CLFS est d'utiliser une partition vide disponible ou, si vous avez assez d'espace non partitionné, d'en créer une. Néanmoins si vous construisez pour une architecture différente, vous pouvez simplement tout construire dans « /mnt/clfs » et le transférer vers votre machine cible.

Un système minimal requiert une partition d'environ 6 Gio (giga octets informatique). C'est suffisant pour conserver toutes les archives tar des sources et pour compiler tous les paquets. Le système CLFS lui-même ne prendra pas autant de place. Une grande partie de cet espace est requise pour fournir temporairement un espace libre suffisant. Compiler des paquets peut demander beaucoup d'espace disque qui sera récupéré après l'installation du paquet. Si le système CLFS a pour but d'être un système Linux primaire, des logiciels supplémentaires seront probablement installés et réclameront une place supplémentaire (entre 2 et 10 Gio).

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 CLFS peut être la même que celle utilisée par le système hôte. Il n'est donc 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 man `cfdisk(8)` ou `fdisk(8)` si vous ne savez pas encore utiliser ces programmes.

Rappelez-vous de la désignation de la nouvelle partition (par exemple hda5). Ce livre y fera référence en tant que la partition CLFS. Rappelez-vous aussi de la désignation de la partition swap. Ces noms seront nécessaires après pour 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 (`ext2`), mais avec les nouveaux disques haute capacité, les systèmes de fichiers journalisés deviennent de plus en plus populaires. Nous allons créer un système de fichiers `ext2`. Les instructions de construction d'autres systèmes de fichiers sont disponibles dans http://cblfs.cross-lfs.org/index.php?section=6#File_System.

Pour créer un système de fichiers `ext2` sur la partition CLFS, lancez ce qui suit :

```
mke2fs /dev/[xxx]
```

Remplacez `[xxx]` par le nom de la partition CLFS (`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 système CLFS au chapitre 9 car toutes ces fonctionnalités ne seront pas supportées par la version d'e2fsprogs installée par CLFS ; 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 `dir_index`, `filetype`, `large_file`, `resize_inode` ou `sparse_super`, 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-crée le système de fichiers sur votre partition CLFS :

```
cd /tmp
tar xjf /path/to/sources/e2fsprogs-1.42.6.tar.bz2
cd e2fsprogs-1.42.6
mkdir build
cd build
../configure
make #Remarquez que nous n'exécutons pas intentionnellement "make install"
./misc/mke2fs /dev/[xxx]
cd /tmp
rm -rf e2fsprogs-1.42.6
```

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

```
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/clfs`, mais le choix du répertoire vous appartient.

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

```
export CLFS=/mnt/clfs
```

Puis, créez le point de montage et montez le système de fichiers CLFS en lançant :

```
mkdir -pv ${CLFS}
mount -v /dev/[xxx] ${CLFS}
```

Remplacez `[xxx]` par la désignation de la partition CLFS.

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

```
mkdir -pv ${CLFS}
mount -v /dev/[xxx] ${CLFS}
mkdir -v ${CLFS}/usr
mount -v /dev/[yyy] ${CLFS}/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`, `nodev` ou `noatime`). Lancez la commande **mount** sans aucun paramètre pour voir les options de la partition CLFS montée. Si `nosuid`, `nodev`, et/ou `noatime` sont configurées, la partition devra être remontée.

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://cross-lfs.org/files/packages/git/>.

Créez un répertoire nommé `${CLFS}/sources` et utilisez-le pour stocker les sources et les correctifs. Tous les paquets devraient être construits là. Il se peut qu'une construction dans un autre endroit donne des résultats inattendus.

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

```
mkdir -v ${CLFS}/sources
```

Affectez le droit d'écriture et le droit sticky sur ce répertoire, ce qui 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 ${CLFS}/sources
```

Vous pouvez télécharger dans ce répertoire tous les paquets et les correctifs en utilisant les liens des pages suivantes de cette section ou en passant la *liste de téléchargement* à `wget`.

3.2. Tous les paquets

Téléchargez ou procurez-vous autrement les paquets suivants :

- **Autoconf (2.69) - 1,188 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 (1.12.4) - 1,346 Ko :**

Page d'accueil : <http://www.gnu.org/software/automake>

Téléchargement : <http://ftp.gnu.org/gnu/automake/automake-1.12.4.tar.xz>

Somme de contrôle MD5 : 7395a0420ecb5c9bc43e5fcf4824df36

- **Bash (4.2) - 6,848 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) - 28.124 Ko :**

Page d'accueil : <http://sources.redhat.com/binutils>

Téléchargement : <http://ftp.gnu.org/gnu/binutils/binutils-2.23.tar.gz>

Somme de contrôle MD5 : ed58f50d8920c3f1d9cb110d5c972c27

• **Bison (2.6.4) - 1,708 Ko :**

Page d'accueil : <http://www.gnu.org/software/bison>

Téléchargement : <http://ftp.gnu.org/gnu/bison/bison-2.6.4.tar.xz>

Somme de contrôle MD5 : 8b2dc57eb9d2d6de4715d30de6b2ee07

• **Bootscripts for CLFS (2.0-pre2) - 44 Ko :**

Téléchargement : <http://cross-lfs.org/files/bootscripts-cross-lfs-2.0-pre2.tar.xz>

Somme de contrôle MD5 : a396eb6898990d93f7de4bf15dad5544

• **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

• **Pkg-config (lite-0.27.1-1) - 396 Ko :**

Page d'accueil : <http://sourceforge.net/projects/pkgconfiglite>

Téléchargement : <http://sourceforge.net/projects/pkgconfiglite/files/0.27.1-1/pkg-config-lite-0.27.1-1.tar.gz>

Somme de contrôle MD5 : 589448b99b6e073924c1bea88dfc9f38

• **Cloog (0.16.3) - 1,900 Ko :**

Page d'accueil : <http://cloog.org>

Téléchargement : <http://www.bastoul.net/cloog/pages/download/cloog-0.16.3.tar.gz>

Somme de contrôle MD5 : a0f8a241cd1c4f103f8d2c91642b3498

• **Coreutils (8.20) - 5,164 Ko :**

Page d'accueil : <http://www.gnu.org/software/coreutils>

Téléchargement : <http://ftp.gnu.org/gnu/coreutils/coreutils-8.20.tar.xz>

Somme de contrôle MD5 : 3d69af8f561fce512538a9fe85f147ff

• **DejaGNU (1.5) - 564 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

• **DHCPD (5.5.6) - 80 Ko :**

Page d'accueil : <http://roy.marples.name/projects/dhcpd>

Téléchargement : <http://roy.marples.name/downloads/dhcpd/dhcpd-5.5.6.tar.bz2>

Somme de contrôle MD5 : a5c0e43b4e836cfc003437329f6b7982

• **Diffutils (3.2) - 1,124 Ko :**

Page d'accueil : <http://www.gnu.org/software/diffutils>

Téléchargement : <http://ftp.gnu.org/gnu/diffutils/diffutils-3.2.tar.xz>

Somme de contrôle MD5 : 26ff64c332429c830c154be46b393382

• **EGLIBC (2.15) - 10,620 Ko:**

Page d'accueil : <http://www.eglibc.org/home>

Téléchargement : <http://cross-lfs.org/files/eglibc-2.15-r21467.tar.xz>

Somme de contrôle MD5 : f4087281e50843e67da86dd8da3ec9a3

• **E2fsprogs (1.42.6) - 4,500 Ko :**

Page d'accueil : <http://e2fsprogs.sourceforge.net>

Téléchargement : <http://www.kernel.org/pub/linux/kernel/people/tytso/e2fsprogs/v1.42.6/e2fsprogs-1.42.6.tar.xz>

Somme de contrôle MD5 : a75d1fffd3980e1470014da3df309c862

• **Expect (5.45) - 616 Ko :**

Page d'accueil : <http://expect.sourceforge.net>

Téléchargement : <http://downloads.sourceforge.net/project/expect/Expect/5.45/expect5.45.tar.gz>

Somme de contrôle MD5 : 44e1a4f4c877e9ddc5a542dfa7ecc92b

• **File (5.11) - 596 Ko :**

Page d'accueil : <http://www.darwinsys.com/file>

Téléchargement : <ftp://ftp.astron.com/pub/file/file-5.11.tar.gz>

Somme de contrôle MD5 : 16a407bd66d6c7a832f3a5c0d609c27b



Remarque

Il se peut que File (5.11) 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://cross-lfs.org/files/packages/git/>.

• **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) - 1,276 Ko :**

Page d'accueil : <http://flex.sourceforge.net>

Téléchargement : <http://downloads.sourceforge.net/flex/flex-2.5.37.tar.bz2>

Somme de contrôle MD5 : c75940e1fc25108f2a7b3ef42abdae06

• **Gawk (4.0.1) - 1,576 Ko :**

Page d'accueil : <http://www.gnu.org/software/gawk>

Téléchargement : <http://ftp.gnu.org/gnu/gawk/gawk-4.0.1.tar.xz>

Somme de contrôle MD5 : a601b032c39cd982f34272664f8afa49

• **GCC (4.6.3) - 70,312 Ko :**

Page d'accueil : <http://gcc.gnu.org>

Téléchargement : <ftp://gcc.gnu.org/pub/gcc/releases/gcc-4.6.3/gcc-4.6.3.tar.bz2>

Somme de contrôle MD5 : 773092fe5194353b02bb0110052a972e

• **Gettext (0.18.1.1) - 14,788 Ko :**

Page d'accueil : <http://www.gnu.org/software/gettext>

Téléchargement : <http://ftp.gnu.org/gnu/gettext/gettext-0.18.1.1.tar.gz>

Somme de contrôle MD5 : 3dd55b952826d2b32f51308f2f91aa89

• **GMP (5.0.5) - 2,008 Ko :**

Page d'accueil : <http://gmplib.org/>

Téléchargement : <http://ftp.gnu.org/gnu/gmp/gmp-5.0.5.tar.bz2>

Somme de contrôle MD5 : 041487d25e9c230b0c42b106361055fe

• **Grep (2.14) - 1,168 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 (1.21) - 3,776 Ko :**

Page d'accueil : <http://www.gnu.org/software/groff>

Téléchargement : <http://ftp.gnu.org/gnu/groff/groff-1.21.tar.gz>

Somme de contrôle MD5 : 8b8cd29385b97616a0f0d96d0951c5bf

• **Gzip (1.5) - 712 Ko :**

Page d'accueil : <http://www.gzip.org>

Téléchargement : <http://ftp.gnu.org/gnu/gzip/gzip-1.5.tar.xz>

Somme de contrôle MD5 : 2a431e169b6f62f7332ef6d47cc53bae

• **Iana-Etc (2.30) - 204 Ko :**

Page d'accueil : <http://www.archlinux.org/packages/core/any/iana-etc/>

Téléchargement : <http://ftp.cross-lfs.org/pub/clfs/conglomeration/iana-etc/iana-etc-2.30.tar.bz2>

Somme de contrôle MD5 : 3ba3afb1d1b261383d247f46cb135ee8

• **IPRoute2 (3.4.0) - 376 Ko :**

Page d'accueil : <http://www.linuxfoundation.org/collaborate/workgroups/networking/iproute2>

Téléchargement : <http://www.kernel.org/pub/linux/utils/net/iproute2/iproute2-3.4.0.tar.xz>

Somme de contrôle MD5 : 879d3fac4e90809598b2864ec4a0cbf8

• **IPutils (s20101006) - 96 Ko :**

Page d'accueil : <http://www.linuxfoundation.org/en/Net:Iputils>

Téléchargement : <http://www.skbuff.net/iputils/iputils-s20101006.tar.bz2>

Somme de contrôle MD5 : a36c25e9ec17e48be514dc0485e7376c

• **Kbd (1.15.3) - 1,624 Ko :**

Téléchargement : <ftp://devel.altlinux.org/legion/kbd/kbd-1.15.3.tar.gz>

Somme de contrôle MD5 : 8143e179a0f3c25646ce5085e8777200

• **Kmod (10) - 1,104 Ko :**

Page d'accueil : <http://git.kernel.org/?p=utils/kernel/kmod/kmod.git;a=summary>

Téléchargement : <http://www.kernel.org/pub/linux/utils/kernel/kmod/kmod-10.tar.xz>

Somme de contrôle MD5 : e2a883c4df15a50f78a7a61d5b64089f

• **Less (451) - 308 Ko :**

Page d'accueil : <http://www.greenwoodsoftware.com/less>

Téléchargement : <http://www.greenwoodsoftware.com/less/less-451.tar.gz>

Somme de contrôle MD5 : 765f082658002b2b46b86af4a0da1842

• **Libee (0.4.1) - 352 Ko :**

Page d'accueil : <http://www.libee.org/>

Téléchargement : <http://www.libee.org/download/files/download/libee-0.4.1.tar.gz>

Somme de contrôle MD5 : 7bbf4160876c12db6193c06e2badedb2

• **Libestr (0.1.0) - 308 Ko :**

Page d'accueil : <http://sourceforge.net/projects/libestr/>

Téléchargement : <http://sourceforge.net/projects/libestr/files/libestr-0.1.0.tar.gz>

Somme de contrôle MD5 : 1b8fe449cffc259075d327334c93bbda

• **Libtool (2.4.2) - 852 Ko :**

Page d'accueil : <http://www.gnu.org/software/libtool>

Téléchargement : <http://ftp.gnu.org/gnu/libtool/libtool-2.4.2.tar.xz>

Somme de contrôle MD5 : 2ec8997e0c07249eb4cbd072417d70fe

• **Linux (3.4.17) - 65,288 Ko :**

Page d'accueil : <http://www.kernel.org>

Téléchargement : <http://www.kernel.org/pub/linux/kernel/v3.0/linux-3.4.17.tar.xz>

Somme de contrôle MD5 : c89817e8856ec88f84ab6a25cc2f7789

• **M4 (1.4.16) - 1,232 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) - 1,216 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 (1.6g) - 252 Ko :**

Page d'accueil : <http://primates.ximian.com/~flucifredi/man>

Téléchargement : <http://primates.ximian.com/~flucifredi/man/man-1.6g.tar.gz>

Somme de contrôle MD5 : ba154d5796928b841c9c69f0ae376660

• **Man-pages (3.43) - 1,076 Ko :**

Page d'accueil : <http://www.win.tue.nl/~aeb/linux/man>

Téléchargement : <http://www.kernel.org/pub/linux/docs/man-pages/man-pages-3.43.tar.xz>

Somme de contrôle MD5 : 761b823ad353975bb87eadb4a8690069

• **MPC (1.0.1) - 616 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

• **MPFR (3.1.1) - 1,048 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

• **Ncurses (5.9) - 2,764 Ko :**

Page d'accueil : <http://www.gnu.org/software/ncurses>

Téléchargement : <ftp://ftp.gnu.org/pub/gnu/ncurses/ncurses-5.9.tar.gz>

Somme de contrôle MD5 : 8cb9c412e5f2d96bc6f459aa8c6282a1

• **Patch (2.7.1) - 668 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) - 13,424 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

• **PPL (0.12.1) - 14,592 Ko :**

Page d'accueil : <http://bugseng.com/products/ppl/>

Téléchargement : <ftp://ftp.cs.unipr.it/pub/ppl/releases/0.12.1/ppl-0.12.1.tar.bz2>

Somme de contrôle MD5 : 8da3ab9de18e669b7af8c4707817d468

• **Procps (3.2.8) - 280 Ko :**

Page d'accueil : <http://procps.sourceforge.net>

Téléchargement : <http://procps.sourceforge.net/procps-3.2.8.tar.gz>

Somme de contrôle MD5 : 9532714b6846013ca9898984ba4cd7e0

• **Psmisc (22.20) - 428 Ko :**

Page d'accueil : <http://psmisc.sourceforge.net>

Téléchargement : <http://downloads.sourceforge.net/psmisc/psmisc-22.20.tar.gz>

Somme de contrôle MD5 : a25fc99a6dc7fa7ae6e4549be80b401f

• **Readline (6.2) - 2,228 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

• **Rsyslog (6.2.2) - 2,376 Ko :**

Page d'accueil : <http://www.rsyslog.com/>

Téléchargement : <http://www.rsyslog.com/files/download/rsyslog/rsyslog-6.2.2.tar.gz>

Somme de contrôle MD5 : b797b8222d6ea4d5dfa007efe8aafa7f

• **Sed (4.2.1) - 880 Ko :**

Page d'accueil : <http://www.gnu.org/software/sed>

Téléchargement : <http://ftp.gnu.org/gnu/sed/sed-4.2.1.tar.bz2>

Somme de contrôle MD5 : 7d310fbd76e01a01115075c1fd3f455a

• **Shadow (4.1.5.1) - 2,144 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

• **Sysvinit (2.88dsf) - 104 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) - 2,288 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 (8.5.12) - 4,412 Ko :**

Page d'accueil : <http://www.tcl.tk>

Téléchargement : <http://downloads.sourceforge.net/tcl/tcl8.5.12-src.tar.gz>

Somme de contrôle MD5 : 174b2b4c619ba8f96875d8a051917703

• **Texinfo (4.13a) - 2,688 Ko :**

Page d'accueil : <http://www.gnu.org/software/texinfo>

Téléchargement : <http://ftp.gnu.org/gnu/texinfo/texinfo-4.13a.tar.gz>

Somme de contrôle MD5 : 71ba711519209b5fb583fed2b3d86fcb

• **Udev (182) - 676 Ko :**

Page d'accueil : <http://www.kernel.org/pub/linux/utils/kernel/hotplug/udev.html>

Téléchargement : <http://www.kernel.org/pub/linux/utils/kernel/hotplug/udev-182.tar.xz>

Somme de contrôle MD5 : 023877e6cc0d907994b8c648beab542b

• **Util-linux (2.22.1) - 3,124 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-2.22.1.tar.xz>

Somme de contrôle MD5 : 730cf9932531ed09b53a04ca30fcb4c9

- **Vim (7.3) - 8,868 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) - 896 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) - 496 Ko :**

Page d'accueil : <http://www.zlib.net>

Téléchargement : <http://zlib.net/zlib-1.2.7.tar.bz2>

Somme de contrôle MD5 : 2ab442d169156f34c379c968f3f482dd

Taille totale de ces paquets : environ NaN MB

3.3. Paquets supplémentaires pour x86_64 Multilib

- **GRUB (2.00) - 5,020 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

Taille totale de ces paquets : environ NaN MB

3.4. Correctifs nécessaires

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 CLFS :

- **Bash Correctif de la branche Mise à jour - 54,711 KB :**

Téléchargement : http://patches.cross-lfs.org/dev/bash-4.2-branch_update-6.patch

Somme de contrôle MD5 : 23c68ff88198537401d49ab6424b005d

- **Coreutils Correctif Uname - 16 Ko :**

Téléchargement : <http://patches.cross-lfs.org/dev/coreutils-8.20-uname-1.patch>

Somme de contrôle MD5 : d47d2d5dec9b4c0b25329511b6b11edf

- **EGLIBC Corrections - 4 Ko :**

Téléchargement : <http://patches.cross-lfs.org/dev/eglibc-2.15-fixes-1.patch>

Somme de contrôle MD5 : 872128f0f087f2036798680c3b118c65

- **Correctif branche Mise à jour de GCC - 601 Ko :**

Téléchargement : http://patches.cross-lfs.org/dev/gcc-4.6.3-branch_update-2.patch

Somme de contrôle MD5 : e7af1c4a02408aeb25c94ed86c7921d6

- **Iana-Etc Correctif Get - 4 Ko :**

Téléchargement : http://patches.cross-lfs.org/dev/iana-etc-2.30-get_fix-1.patch

Somme de contrôle MD5 : 7da753875d46cab21763dac0db571b3

- **Iana-Etc Correctif Mise à jour des numéros de ports dans Protocol et Services - 3,760 Ko :**

Téléchargement : http://patches.cross-lfs.org/dev/iana-etc-2.30-numbers_update-20120610-2.patch

Somme de contrôle MD5 : 826fb780d13caafb7cb99b9c346f2102

• **IPUtils correctif - 8 Ko :**

Téléchargement : <http://patches.cross-lfs.org/dev/iputils-s20101006-fixes-1.patch>

Somme de contrôle MD5 : 1add4b8cbee814310f95e61997019162

• **IPUtils correctif documentation prégénérée - 136 Ko :**

Téléchargement : <http://patches.cross-lfs.org/dev/iputils-s20101006-doc-1.patch>

Somme de contrôle MD5 : 2eee5e095005bf4be426797a4aefa27b

• **Kbd correction es.po - 4 Ko :**

Téléchargement : http://patches.cross-lfs.org/dev/kbd-1.15.3-es.po_fix-1.patch

Somme de contrôle MD5 : 476c4066c5c663b44b67acaa4cdef62e

• **M4 Correctif pour obtenir - 4 Ko :**

Téléchargement : <http://patches.cross-lfs.org/dev/m4-1.4.16-no-gets-1.patch>

Somme de contrôle MD5 : 6c5013f9ae5afc78f123e96356ceec3e

• **Man correctif i18n - 12 Ko :**

Téléchargement : <http://patches.cross-lfs.org/dev/man-1.6g-i18n-1.patch>

Somme de contrôle MD5 : a5aba0cb5a95a7945db8c882334b7dab

• **Ncurses Correctif Bash - 4 Ko :**

Téléchargement : http://patches.cross-lfs.org/dev/ncurses-5.9-bash_fix-1.patch

Somme de contrôle MD5 : c6f7f2ab0ebaf7721eb266641352db

• **Ncurses Correctif de la branche Mise à jour - 2,492 Ko :**

Téléchargement : http://patches.cross-lfs.org/dev/ncurses-5.9-branch_update-4.patch

Somme de contrôle MD5 : c2b2dc2d31b02c218359e6218f12a72c

• **Perl Correctif Libc - 20 Ko :**

Téléchargement : <http://patches.cross-lfs.org/dev/perl-5.16.2-libc-1.patch>

Somme de contrôle MD5 : 665f85a83b6141776499f792514235c7

• **Procps Correctif erreur HZ - 4 Ko :**

Téléchargement : http://patches.cross-lfs.org/dev/procps-3.2.8-fix_HZ_errors-1.patch

Somme de contrôle MD5 : 2ea4c8e9a2c2a5a291ec63c92d7c6e3b

• **Procps correctif ps cgroup - 4 Ko :**

Téléchargement : http://patches.cross-lfs.org/dev/procps-3.2.8-ps_cgroup-1.patch

Somme de contrôle MD5 : 3c478ef88fad23353e332b1b850ec630

• **Readline Correctif Branche Mise à jour - 4 Ko :**

Téléchargement : http://patches.cross-lfs.org/dev/readline-6.2-branch_update-3.patch

Somme de contrôle MD5 : af788f5b1cfc5db9efc9e0fa0268a574

• **Tar correctif pages de man - 76 Ko :**

Téléchargement : <http://patches.cross-lfs.org/dev/tar-1.26-man-1.patch>

Somme de contrôle MD5 : 074783d41f18c5c62a7cfc77e2678693

• **Texinfo correctif nouveaux outils de compression - 4 Ko :**

Téléchargement : http://patches.cross-lfs.org/dev/texinfo-4.13a-new_compressors-1.patch

Somme de contrôle MD5 : 4ae2d3c132e21cb83b825bc691056d07

• **Vim Correctif de la branche Mise à jour - 2,980 Ko :**

Téléchargement : http://patches.cross-lfs.org/dev/vim-7.3-branch_update-6.patch

Somme de contrôle MD5 : 21cfe3150e5316ef272012630950b7ad

Taille totale de ces correctifs : environ NaN MB

En plus des correctifs requis ci-dessus, il existe un certain nombre de correctifs optionnels créés par la communauté CLFS. 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://patches.cross-lfs.org/dev/> et vous pouvez récupérer tout correctif supplémentaire correspondant aux besoins de votre système.

3.5. Correctifs supplémentaires pour x86_64 multilib

- **GCC Specs Patch - 20 Ko:**

Téléchargement : <http://patches.cross-lfs.org/dev/gcc-4.6.3-specs-1.patch>

Somme de contrôle MD5 : 61d583984f9f12b6f37141e132fc7d57

- **IPRoute2 correctif Lib64 - 2.0 Ko :**

Téléchargement : <http://patches.cross-lfs.org/dev/iproute2-3.4.0-libdir-1.patch>

Somme de contrôle MD5 : cf8948c05f3a641912e6bd1b38a8a382

- **Perl correctif Configure Multilib - 560 Ko :**

Téléchargement : http://patches.cross-lfs.org/dev/perl-5.16.2-Configure_multilib-1.patch

Somme de contrôle MD5 : f3e57e768d985b03e93848eb401e8ab4

Taille totale de ces correctifs : environ NaN MB

Chapitre 4. Dernières préparations

4.1. À propos de `${CLFS}`

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

```
echo ${CLFS}
```

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

```
export CLFS=/mnt/lfs
```

Avoir cette variable initialisée est bénéfique car des commandes telles que **install -dv `${CLFS}/tools`** peuvent être saisies de façon littérale. Votre shell remplacera « `${CLFS}` » par « `/mnt/clfs` » (ou par ce avec quoi vous avez initialisé la variable) lorsqu'il exécutera la ligne de commande.

Si vous n'avez pas créé le répertoire `${CLFS}`, faites-le maintenant en lançant les commandes suivantes :

```
install -dv ${CLFS}
```

N'oubliez pas de vérifier que `${CLFS}` 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 `${CLFS}/tools`

Tous les programmes compilés dans *Constructing a Temporary System* seront installés dans `${CLFS}/tools` pour les tenir séparés des programmes compilés dans le *Installing Basic System Software*. Les programmes compilés ici sont seulement des outils temporaires et ne prendront pas part au système `CLFS` final. En les conservant dans un répertoire séparé, nous pourrions 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 *Constructing a Temporary System*).

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

```
mkdir -v ${CLFS}/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 `CLFS`. Lancez cette commande en tant qu'utilisateur `root` :

```
ln -sv ${CLFS}/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. Cela fournira un répertoire commun pour nos outils temporaires.

4.3. Créer le répertoire `${CLFS}/cross-tools`

Le binutils et le compilateur croisés construits dans le Constructing Cross-Compile Tools seront installées sous `${CLFS}/cross-tools` pour les tenir séparés des programmes de l'hôte. Les programmes construits ici sont des outils croisés et ne feront pas partie du système CLFS final ou du système temporaire. En laissant ces programmes dans un répertoire séparé, vous pouvez facilement les supprimer plus tard après leur utilisation.

Créez le répertoire nécessaire en lançant ce qui suit en tant qu'utilisateur `root` :

```
install -dv ${CLFS}/cross-tools
```

La prochaine étape consiste en la création du lien symbolique `/cross-tools` sur le système hôte. Il va pointer vers le répertoire récemment créé sur la partition CLFS. Lancez cette commande en tant qu'utilisateur `root` :

```
ln -sv ${CLFS}/cross-tools /
```

Techniquement, le lien symbolique n'est pas nécessaire (bien que les instructions du livre supposent qu'il existe) mais il est principalement là par cohérence (parce que `/tools` est aussi lié de manière symbolique à `${CLFS}/tools`) et pour simplifier l'installation des outils de compilation croisée.

4.4. Ajouter l'utilisateur CLFS

Lorsque vous êtes connecté en tant qu'utilisateur `root`, faire une seule erreur peut endommager voire détruire 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 `clfs` comme membre d'un nouveau groupe `clfs` utilisez-le lors du processus d'installation. En tant que `root`, lancez les commandes suivantes pour créer le nouvel utilisateur :

```
groupadd clfs
useradd -s /bin/bash -g clfs -d /home/clfs clfs
mkdir -pv /home/clfs
chown -v clfs:clfs /home/clfs
```

Voici la signification des options en ligne de commande :

`-s /bin/bash`

Ceci fait de **bash** le shell par défaut de l'utilisateur `clfs`.



Important

Les instructions de construction supposent que vous utilisez le shell **bash**.

`-g clfs`

Cette option ajoute l'utilisateur `clfs` au groupe `clfs`.

`clfs`

Ceci est le nom réel du groupe et de l'utilisateur créé.

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

```
passwd clfs
```

Donnez à `clfs` un accès complet à `${CLFS}/cross-tools` et à `${CLFS}/tools` en indiquant que `clfs` est le propriétaire des répertoires :

```
chown -v clfs ${CLFS}/tools
chown -v clfs ${CLFS}/cross-tools
```

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

```
chown -v clfs ${CLFS}/sources
```

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

```
su - clfs
```

Le « - » indique à `su` un shell de connexion par opposition à un shell de non connexion. Vous trouverez la différence entre les deux types de shells dans la page `man bash(1)` et **info bash**.



Remarque

Tant que rien d'autre n'est indiqué, toutes les commandes à partir de maintenant se lancent en tant qu'utilisateur `clfs`.

4.5. 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 `clfs`, 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 `clfs`, 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 réalise le but d'avoir un 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
CLFS=/mnt/clfs
LC_ALL=POSIX
PATH=/cross-tools/bin:/bin:/usr/bin
export CLFS LC_ALL 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 `/cross-tools` dès qu'ils sont disponibles et sans se rappeler d'une 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 `CLFS` 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 autre chose 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.

En plaçant `/cross-tools/bin` au début de `PATH`, le compilateur croisé construit dans Constructing Cross-Compile Tools sera choisi par le processus de construction du système temporaire avant tout programme installé sur l'hôte. Ceci, combiné avec la désactivation du hachage, permet de s'assurer que vous utiliserez les outils de compilation croisée pour construire le système temporaire dans `/tools`.

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
```

4.6. 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 comme le développeur en avait l'intention. Néanmoins, cela ne garantit pas que le paquet ne contient pas de bogues.

Il n'est pas possible de lancer les suites de tests lors d'une compilation croisée, donc les instructions d'installation des paquets n'expliquent pas comment lancer les suites de tests jusqu'au Installing Basic System Software.

Partie III. Fabriquer les outils de compilation croisée

Chapitre 5. Construire les outils de compilation croisée

5.1. Introduction

Ce chapitre vous montre comment créer des outils pour une plateforme croisée.

Si, pour quelque raison que ce soit, vous devez vous arrêter et revenir plus tard, rappelez-vous d'utiliser la commande **su - cifs** et elle initialisera l'environnement de construction que vous avez quitté.

5.1.1. Remarques générales



Important

Avant d'exécuter les instructions de construction pour un paquet, vous devriez déballer le paquet et effectuer un **cd** dans le répertoire créé.

Plusieurs paquets sont corrigés avant d'être compilés, mais seulement quand le correctif est nécessaire pour contourner un problème. Un correctif est souvent nécessaire à la fois dans ce chapitre et dans les suivants, mais parfois uniquement dans un des autres. Donc, ne vous affolez pas si les instructions pour un correctif téléchargé vous paraissent absentes. Il se peut que vous rencontriez des messages d'avertissement concernant *offset* ou *fuzz* lorsque vous appliquerez un correctif. Ne vous en inquiétez pas car le correctif a été appliqué avec succès.

Pendant la compilation de la plupart des paquets, il y aura plusieurs avertissements qui vont défiler sur l'écran. Ils sont normaux et vous pouvez les ignorer en toute sécurité. Comme ils le disent, ces messages sont des—avertissements concernant l'usage d'une syntaxe C ou C++ obsolète ou invalide. Les standards du C changent assez souvent et certains paquets utilisent encore de vieux standards. Cela n'est pas un problème mais affiche des avertissements.



Important

Après l'installation de chaque paquet, qu'il s'agisse de ce chapitre ou des suivants, effacez ses répertoires de sources et de construction sauf si on vous indique autre chose. L'effacement des sources empêche une mauvaise configuration quand vous réinstallerez le même paquet plus tard.

5.2. CFLAGS de construction

CFLAGS et CXXFLAGS ne doivent pas être initialisées pendant la construction des outils croisés.

Pour désactiver CFLAGS et CXXFLAGS utilisez les commandes suivantes :

```
unset CFLAGS
unset CXXFLAGS
```

Maintenant, ajoutez ce qui suit à `~/ .bashrc`, pour le cas où vous devez quitter et redémarrer la construction plus tard :

```
echo unset CFLAGS >> ~/ .bashrc
echo unset CXXFLAGS >> ~/ .bashrc
```

5.3. Variables de construction

Initialiser l'hôte et la cible

Pendant la construction des outils de compilation croisée, vous aurez besoin de régler quelques variables en fonction de vos besoins particuliers. La première variable sera le triplet de la machine hôte, qui sera contenu dans la variable `CLFS_HOST`. Pour prendre en compte la possibilité que l'hôte et la cible aient la même architecture, étant donné que la compilation croisée ne fonctionnera pas lorsque l'hôte et la cible sont les mêmes, il faudra modifier légèrement une partie du triplet pour ajouter "cross". Réglez `CLFS_HOST` en utilisant la commande suivante :

```
export CLFS_HOST=$(echo ${MACHTYPE} | sed -e 's/-[^-]*/-cross/')
```

Maintenant, vous devrez paramétrer le triplet pour l'architecture cible. Paramétrez la variable cible en utilisant la commande suivante :

```
export CLFS_TARGET="x86_64-unknown-linux-gnu"
```

Maintenant, nous allons régler notre triplette cible pour 32 bits :

```
export CLFS_TARGET32="i686-pc-linux-gnu"
```

Copier les paramètres vers l'environnement

Ajoutez maintenant ceux-ci à `~/ .bashrc`, au cas où vous devriez quitter et recommencer la construction plus tard :

```
cat >> ~/.bashrc << EOF
export CLFS_HOST="${CLFS_HOST}"
export CLFS_TARGET="${CLFS_TARGET}"
export CLFS_TARGET32="${CLFS_TARGET32}"
EOF
```

5.4. Options de compilation

Nous allons devoir mettre en place des drapeaux de compilation spécifiques destinés au compilateur et à l'éditeur de liens :

```
export BUILD32="-m32"
export BUILD64="-m64"
```

Ajoutons les drapeaux de compilation au fichier `~/ .bashrc` afin d'éviter toute erreur si vous deviez quitter l'environnement et revenir plus tard :

```
cat >> ~/.bashrc << EOF
export BUILD32="${BUILD32}"
export BUILD64="${BUILD64}"
EOF
```


5.5. Linux-Headers-3.4.17

Le noyau Linux contient une cible make qui installe des en-têtes du noyau « propres ».

5.5.1. Installation de Linux-Headers

Pour cette étape, vous aurez besoin de l'archive tar du noyau.

Installez les fichiers d'en-tête du noyau :

```
install -dv /tools/include
make mrproper
make ARCH=x86_64 headers_check
make ARCH=x86_64 INSTALL_HDR_PATH=dest headers_install
cp -rv dest/include/* /tools/include
```

Voici la signification des commandes :

make mrproper

S'assure que le répertoire des sources du noyau est propre.

make ARCH=x86_64 headers_check

Nettoie les en-têtes raw du noyau afin qu'elles puissent être utilisées par les programmes d'espace utilisateur.

make ARCH=x86_64 INSTALL_HDR_PATH=dest headers_install

La cible `headers_install` supprime normalement tout le répertoire de destination (par défaut `/usr/include`) avant d'installer les en-têtes. Pour empêcher cela, nous disons au noyau d'installer les en-têtes dans un autre répertoire à l'intérieur de celui des sources.

Les détails sur ce paquet sont situés dans Section 10.5.2, « Contenu de Linux-Headers. »

5.6. File-5.11

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

5.6.1. Installation de File

Préparez la compilation de File :

```
./configure --prefix=/cross-tools --disable-static
```

Voici la signification des options de configure :

--prefix=/cross-tools

Ceci dit au script configure de se préparer à installer le paquet dans le répertoire */cross-tools*.

--disable-static

Ceci dit au paquet File de ne pas compiler ou installer les bibliothèques statiques qui ne sont pas utiles pour les outils croisés

Compilez le paquet :

```
make
```

Installez le paquet :

```
make install
```

Les détails de ce paquet sont situés dans Section 10.57.2, « Contenu de File. »

5.7. M4-1.4.16

Le paquet M4 contient un processeur de macros.

5.7.1. Installation de M4

Le correctif suivant contient une correction lors de la construction avec un hôte ayant Glibc ou EGLIBC 2.16 ou supérieur.

```
patch -Np1 -i ../m4-1.4.16-no-gets-1.patch
```

Préparez la compilation de M4 :

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

Compilez le paquet :

```
make
```

Installez le paquet :

```
make install
```

Les détails sur ce paquet sont disponibles dans Section 10.38.2, « Contenu de M4. »

5.8. Ncurses-5.9

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

5.8.1. Installation de Ncurses

Le correctif suivant corrige des problèmes avec certaines versions de Bash :

```
patch -Np1 -i ../ncurses-5.9-bash_fix-1.patch
```

Prepare Ncurses for compilation:

```
./configure --prefix=/cross-tools \
  --without-debug --without-shared
```

Voici la signification des nouvelles options de configure :

--without-debug

Dit à Ncurses de se construire sans les informations de débogage.

--without-shared

Ceci empêche Ncurses de construire ses bibliothèques partagées qui ne sont pas utiles pour l'instant.

Un seul binaire est nécessaire pour les outils de compilation croisée. Construisez les en-têtes puis construisez **tic** :

```
make -C include
make -C progs tic
```

Installez **tic** avec la commande suivante :

```
install -v -m755 progs/tic /cross-tools/bin
```

Les détails sur ce paquet sont disponibles dans Section 10.27.2, « Contenu de Ncurses. »

5.9. GMP-5.0.5

GMP est une bibliothèque pour faire de l'arithmétique en précision arbitraire sur les entiers, les nombres rationnels et les nombres flottants.

5.9.1. Installation de GMP

Préparez la compilation de GMP :

```
CPPFLAGS=-fexceptions ./configure \  
--prefix=/cross-tools --enable-cxx --disable-static
```

Voici la signification des nouvelles options de configure :

CPPFLAGS=-fexceptions

Permet à GMP de gérer les exceptions C++ prises par PPL.

--enable-cxx

Ceci dit à GMP d'activer le support de C++.

Compilez le paquet :

```
make
```

Installez le paquet :

```
make install
```

Les détails sur ce paquet sont disponibles dans Section 10.11.2, « Contenu de GMP. »

5.10. MPFR-3.1.1

La bibliothèque MPFR est une bibliothèque C pour des calculs de nombres flottants à précision multiple avec un arrondis correct.

5.10.1. Installation de MPFR

Préparez la compilation de MPFR :

```
LDFLAGS="-Wl,-rpath,/cross-tools/lib" \
./configure --prefix=/cross-tools \
--enable-shared --disable-static --with-gmp=/cross-tools
```

Voici la signification des nouvelles options de configure :

LDFLAGS="-Wl,-rpath,/cross-tools/lib"

Ceci dit à **configure** de chercher les bibliothèques dans */cross-tools*.

--enable-shared

Ceci dit à **configure** de construire les bibliothèques partagées de MPFR.

--with-gmp=/cross-tools

Ceci dit à **configure** où trouver GMP.

Compilez le paquet :

```
make
```

Installez le paquet :

```
make install
```

Les détails sur ce paquet sont disponibles dans Section 10.13.2, « Contenu de MPFR. »

5.11. MPC-1.0.1

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

5.11.1. Installation de MPC

Préparez la compilation de MPC :

```
LDFLAGS="-Wl,-rpath,/cross-tools/lib" \  
./configure --prefix=/cross-tools --disable-static \  
--with-gmp=/cross-tools --with-mpfr=/cross-tools
```

Compilez le paquet :

```
make
```

Installez le paquet :

```
make install
```

Les détails sur ce paquet sont disponibles dans Section 10.15.2, « Contenu de MPC. »

5.12. PPL-0.12.1

La bibliothèque *Parma Polyhedra Library* (PPL) fournit des abstractions numériques destinées principalement à des applications dans le domaine de l'analyse et de la vérification de systèmes complexes. CLooG-PPL exige cette bibliothèque.

5.12.1. Installation de PPL

Préparez la compilation de PPL :

```
CPPFLAGS="-I/cross-tools/include" \
  LDFLAGS="-Wl,-rpath,/cross-tools/lib" \
  ./configure --prefix=/cross-tools --enable-shared --disable-static \
  --enable-interfaces="c,cxx" --disable-optimization \
  --with-gmp=/cross-tools
```

Voici la signification de la nouvelle option de configure :

`--enable-interfaces="c,cxx"`

Dit à **configure** d'activer le support de C et de C++.

`--disable-optimization`

Dit à **configure** de construire PPL sans les optimisations du compilateur qui ne sont pas nécessaires pour les outils croisés.

Compilez le paquet :

```
make
```

Installez le paquet :

```
make install
```

Les détails sur ce paquet sont disponibles dans Section 10.17.2, « Contenu de PPL. »

5.13. CLooG-0.16.3

CLooG est une bibliothèque pour générer du code pour analyser des polyèdres Z. En d'autres termes, il trouve du code qui atteint chaque point entier (ou intégral) d'un ou plusieurs polyèdres paramétrés. GCC se lie à cette bibliothèque afin d'activer le nouveau code de génération de boucle, connu en tant que Graphite.

5.13.1. Installation de CLooG

Ce qui suit empêche le script configure de définir LD_LIBRARY_PATH quand il trouve PPL. Cela empêchera tout any conflit avec des bibliothèques du système hôte :

```
cp -v configure{,.orig}
sed -e "/LD_LIBRARY_PATH=/d" \
    configure.orig > configure
```

Prepare CLooG for compilation:

```
LDFLAGS="-Wl,-rpath,/cross-tools/lib" \
. ./configure --prefix=/cross-tools --enable-shared --disable-static \
--with-gmp-prefix=/cross-tools
```

Compilez le paquet :

```
make
```

Installez le paquet :

```
make install
```

Les détails sur ce paquet sont disponibles dans Section 10.19.2, « Contenu de CLooG. »

5.14. Binutils-2.23 croisé

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

5.14.1. Installation de Binutils croisé

Il est important que Binutils soit construit avant Glibc et GCC car les deux effectuent divers tests sur l'éditeur de liens et l'assembleur disponibles pour déterminer quelles fonctionnalités activer.

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 :

```
AR=ar AS=as ../binutils-2.23/configure \
  --prefix=/cross-tools --host=${CLFS_HOST} --target=${CLFS_TARGET} \
  --with-sysroot=${CLFS} --with-lib-path=/tools/lib --disable-nls \
  --enable-shared --disable-static --enable-64-bit-bfd
```

Voici la signification des options de configure :

AR=ar AS=as

Ceci empêche Binutils de se compiler avec `${CLFS_HOST}-ar` et `${CLFS_HOST}-as` car ils sont fournis par ce paquet et ils ne peuvent donc pas encore être installés.

--host=\${CLFS_HOST}

Lors d'une utilisation avec `--target`, ceci crée un exécutable pour une architecture croisée qui crée des fichiers pour `${CLFS_TARGET}` mais s'exécute sur `${CLFS_HOST}`.

--target=\${CLFS_TARGET}

Lors d'une utilisation avec `--host`, ceci crée un exécutable pour une architecture croisée qui crée des fichiers pour `${CLFS_TARGET}` mais se lance sur `${CLFS_HOST}`.

--with-lib-path=/tools/lib

Ceci dit au script configure de spécifier le chemin de recherche de la bibliothèque pendant la compilation de Binutils, le résultat dans `/tools/lib` étant passé à l'éditeur de liens. Ceci empêche l'éditeur de liens de chercher à travers les répertoires de la bibliothèque sur l'hôte.

--disable-nls

Ceci désactive l'internationalisation car `i18n` n'est pas nécessaire pour les outils de compilation croisée.

--disable-multilib

Cette option désactive la construction d'un Binutils supportant le Multilib.

--enable-64-bit-bfd

Ceci ajoute le support pour 64 bit à Binutils.

Compilez le paquet :

```
make configure-host
make
```

Voici la signification des options de make :

configure-host

Ceci vérifie l'environnement hôte et s'assure que tous les outils nécessaires sont disponibles pour compiler Binutils.

Installez le paquet :

```
make install
```

Copiez `libiberty.h` vers le répertoire `/tools/include` :

```
cp -v ../binutils-2.23/include/libiberty.h /tools/include
```

Les détails sur ce paquet sont disponibles dans Section 10.22.2, « Contenu de Binutils. »

5.15. GCC-4.6.3 croisé - Statique

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

5.15.1. Installation du compilateur GCC croisé avec libgcc statique et sans gestion des « Threads »

Le correctif suivant contient un certain nombre de mises à jour vers la branche 4.6.3, faites par les développeurs de GCC :

```
patch -Np1 -i ../gcc-4.6.3-branch_update-2.patch
```

Faites deux ajustements essentiels pour le fichier `specs` de GCC pour vous assurer que GCC utilise notre environnement de construction :

```
patch -Np1 -i ../gcc-4.6.3-specs-1.patch
```

Modifiez la spec `StartFile` et le `Standard Include Dir` afin que GCC regarde dans `/tools` :

```
echo -en '#undef STANDARD_INCLUDE_DIR\n#define STANDARD_INCLUDE_DIR "/tools/include/'
echo -en '\n#define STANDARD_STARTFILE_PREFIX_1\n#define STANDARD_STARTFILE_PREFIX_2\n'
echo -en '\n#define STANDARD_STARTFILE_PREFIX_2\n#define STANDARD_STARTFILE_PREFIX_2\n'
```

Maintenant, modifiez le chemin de recherche include par défaut du préprocesseur c de GCC pour n'utiliser que `/tools` :

```
cp -v gcc/Makefile.in{,.orig}
sed -e "s@(^CROSS_SYSTEM_HEADER_DIR =\).*@1 /tools/include@g" \
    gcc/Makefile.in.orig > gcc/Makefile.in
```

Nous allons créer un faux `limits.h` pour que la construction n'utilise pas celui fourni par la distrib hôte :

```
touch /tools/include/limits.h
```

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 :

```
AR=ar LDFLAGS="-Wl,-rpath,/cross-tools/lib" \
../gcc-4.6.3/configure --prefix=/cross-tools \
--build=${CLFS_HOST} --host=${CLFS_HOST} --target=${CLFS_TARGET} \
--with-sysroot=${CLFS} --with-local-prefix=/tools --disable-nls \
--disable-shared --with-mpfr=/cross-tools \
--with-gmp=/cross-tools --with-ppl=/cross-tools --with-cloog=/cross-tools \
--without-headers --with-newlib --disable-decimal-float \
--disable-libgomp --disable-libmudflap --disable-libssp \
--disable-threads --enable-languages=c --enable-cloog-backend=isl
```

Voici la signification des nouvelles options de configure :

```
--with-sysroot=${CLFS}
```

Dit à GCC de considérer `${CLFS}` comme le système de fichiers racine.

`--with-local-prefix=/tools`

Le but de ce paramètre est de supprimer `/usr/local/include` du chemin de recherche include de **gcc**. Ce n'est pas absolument essentiel, néanmoins cela aide à minimiser l'influence du système hôte.

`--disable-nls`

Ceci désactive l'internationalisation car l'i18n n'est pas nécessaire pour les outils de compilation croisée.

`--without-headers`

Désactive l'utilisation par GCC de la Libc de la cible lors de la compilation croisée.

`--with-newlib`

Dit à GCC que la libc cible utilisera 'newlib'.

`--disable-decimal-float`

Désactive le support de l'extension des points flottants décimaux en C.

`--disable-libgomp`

Désactive la création de bibliothèques utilisées au moment de l'exécution de GOMP.

`--disable-libmudflap`

Désactive la création des bibliothèques utilisées au moment de l'exécution par libmudflap.

`--disable-libssp`

Désactive l'utilisation de *Stack Smashing Protection* (protection du smashing de la pile) pour les bibliothèques utilisées au moment d'une exécution.

`--disable-threads`

Cela empêchera GCC de chercher les fichiers include multi-thread, vu qu'ils n'ont pas encore été créés pour cette architecture. GCC sera capable de trouver les informations multi-thread après que les en-têtes Glibc ont été créés.

`--enable-languages=c`

Cette option nous assure que seul le compilateur C sera construit.

Poursuivez en compilant le paquet :

```
make all-gcc all-target-libgcc
```

Voici la signification des nouvelles options de **make** :

`all-gcc all-target-libgcc`

Ne compile que les parties de GCC nécessaires pour l'instant, et non tout le paquet.

Installez le paquet :

```
make install-gcc install-target-libgcc
```

Les détails sur ce paquet sont situés dans Section 10.23.2, « Contenu de GCC. »

5.16. EGLIBC-2.15 32 bit

Le paquet EGLIBC contient la bibliothèque C principale. Cette bibliothèque fournit toutes les routines de base pour allouer de la mémoire, rechercher dans des répertoires, ouvrir et fermer des fichiers, les lire et les écrire, gérer les chaînes, faire de la recherche de motifs, faire de l'arithmétique etc.

5.16.1. Installation de EGLIBC

Remarquez que toute autre méthode de construction Glibc que celle suggérée dans ce livre met en péril la stabilité du système.

Désactivez l'édition d'un lien vers `libgcc_eh`:

```
cp -v Makeconfig{,.orig}
sed -e 's/-lgcc_eh//g' Makeconfig.orig > Makeconfig
```

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

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

Vous devez ajouter les lignes suivantes à `config.cache` pour qu'EGLIBC supporte NPTL :

```
cat > config.cache << "EOF"
libc_cv_forced_unwind=yes
libc_cv_c_cleanup=yes
libc_cv_gnu89_inline=yes
libc_cv_ssp=no
EOF
```

Préparez la compilation d'EGLIBC :

```
BUILD_CC="gcc" CC="${CLFS_TARGET}-gcc ${BUILD32}" \
  AR="${CLFS_TARGET}-ar" RANLIB="${CLFS_TARGET}-ranlib" \
  CFLAGS="-march=$(cut -d- -f1 <<< ${CLFS_TARGET}32) -mtune=generic -g -O2" \
  ../eglibc-2.15/configure --prefix=/tools \
  --host=${CLFS_TARGET32} --build=${CLFS_HOST} \
  --disable-profile --with-tls --enable-kernel=2.6.32 --with-__thread \
  --with-binutils=/cross-tools/bin --with-headers=/tools/include \
  --cache-file=config.cache
```

Voici la signification des nouvelles options de `configure` :

BUILD_CC="gcc"

Ceci règle GCC pour qu'il utilise le compilateur actuel de notre système. On l'utilise pour créer les outils dont Glibc se sert pendant sa construction.

CC="\${CLFS_TARGET}-gcc \${BUILD32}"

Cette option force EGLIBC à utiliser notre architecture cible avec des drapeaux 32 bits.

AR="\${CLFS_TARGET}-ar"

Ceci oblige Glibc à utiliser l'outil **ar** que nous avons construit pour notre architecture cible.

RANLIB="\${CLFS_TARGET}-ranlib"

Ceci oblige Glibc à utiliser l'outil **ranlib** que nous avons construit pour notre architecture cible.

```
CFLAGS="-march=$(cut -d- -f1 <<< $CLFS_TARGET32) -mtune=generic -g -O2"
```

Oblige EGLIBC à s'optimiser pour notre système cible.

```
--disable-profile
```

Ceci construit les bibliothèques sans informations de profilage. N'utilisez pas cette option si le profiling est nécessaire sur les outils temporaires.

```
--with-tls
```

Ceci dit à Glibc d'utiliser Thread Local Storage.

```
--enable-kernel=2.6.32
```

Ceci dit à Glibc de compiler la bibliothèque avec le support pour les noyaux Linux 2.6.32 et supérieurs.

```
--with-__thread
```

Ceci dit à Glibc d'utiliser `__thread` pour la construction de `libc` et de `libpthread`.

```
--with-binutils=/cross-tools/bin
```

Ceci dit à Glibc d'utiliser les Binutils spécifiques à notre architecture cible.

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

Ceci dit à Glibc de se compiler avec les en-têtes récemment installées dans le répertoire `/tools`, afin qu'il sache exactement quelles fonctionnalités a le noyau et qu'il puisse s'optimiser en conséquence.

```
--cache-file=config.cache
```

Ceci dit à Glibc d'utiliser un fichier de cache préfabriqué.

Pendant cette étape, il se pourrait que les avertissements suivants apparaissent :

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

L'absence ou l'incompatibilité du programme **msgfmt** n'est en général pas gênant. Ce programme **msgfmt** fait partie du paquet `Gettext` que la distribution hôte devrait fournir.

Compilez le paquet :

```
make
```

Installez le paquet :

```
make install inst_varbasedir=/tools/var/db
```

Les détails sur ce paquet sont situés dans Section 10.8.5, « Contenu d'EGLIBC. »

5.17. EGLIBC-2.15 64 bits

Le paquet EGLIBC contient la bibliothèque C principale. Cette bibliothèque fournit toutes les routines de base pour allouer de la mémoire, rechercher dans des répertoires, ouvrir et fermer des fichiers, les lire et les écrire, gérer les chaînes, faire de la recherche de motifs, faire de l'arithmétique etc.

5.17.1. Installation de EGLIBC

Remarquez que toute autre méthode de construction Glibc que celle suggérée dans ce livre met en péril la stabilité du système.

Désactivez l'édition d'un lien vers `libgcc_eh`:

```
cp -v Makeconfig{,.orig}
sed -e 's/-lgcc_eh//g' Makeconfig.orig > Makeconfig
```

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

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

Vous devez ajouter les lignes suivantes à `config.cache` pour qu'EGLIBC supporte NPTL :

```
cat > config.cache << "EOF"
libc_cv_forced_unwind=yes
libc_cv_c_cleanup=yes
libc_cv_gnu89_inline=yes
libc_cv_ssp=no
EOF
```

Ordonnez à EGLIBC d'installer ses bibliothèques 64 bits dans `/tools/lib64` :

```
echo "slibdir=/tools/lib64" >> configparms
```

Préparez la compilation d'EGLIBC :

```
BUILD_CC="gcc" CC="${CLFS_TARGET}-gcc ${BUILD64}" \
  AR="${CLFS_TARGET}-ar" RANLIB="${CLFS_TARGET}-ranlib" \
  ../eglibc-2.15/configure --prefix=/tools \
  --host=${CLFS_TARGET} --build=${CLFS_HOST} --libdir=/tools/lib64 \
  --disable-profile --with-tls --enable-kernel=2.6.32 --with-__thread \
  --with-binutils=/cross-tools/bin --with-headers=/tools/include \
  --cache-file=config.cache
```

Voici la signification des options de configure :

```
CC="${CLFS_TARGET}-gcc ${BUILD64}"
```

Force EGLIBC à compiler en utilisant le GCC de notre architecture cible avec des drapeaux 64 bits.

```
--libdir=/tools/lib64
```

Installe EGLIBC dans `/tools/lib64` plutôt que dans `/tools/lib`.

Pendant cette étape, il se pourrait que les avertissements suivants apparaissent :

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

L'absence ou l'incompatibilité du programme **msgfmt** n'est en général pas gênant. Ce programme **msgfmt** fait partie du paquet Gettext que la distribution hôte devrait fournir.

Compilez le paquet :

```
make
```

Installez le paquet :

```
make install inst_varbdir=/tools/var/db
```

Installez les en-têtes liées à NIS et RPC qui ne sont pas installées par défaut.

```
cp -v ../eglibc-2.15/sunrpc/rpc/*.h /tools/include/rpc
cp -v ../eglibc-2.15/sunrpc/rpcsvc/*.h /tools/include/rpcsvc
cp -v ../eglibc-2.15/nis/rpcsvc/*.h /tools/include/rpcsvc
```

Les détails sur ce paquet sont situés dans Section 10.8.5, « Contenu d'EGLIBC. »

5.18. GCC-4.6.3 croisé - compilation finale

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

5.18.1. Installation du compilateur GCC croisé

Le correctif suivant contient un certain nombre de mises à jour vers la branche 4.6.3, faites par les développeurs de GCC :

```
patch -Np1 -i ../gcc-4.6.3-branch_update-2.patch
```

Faites deux ajustements essentiels pour le fichier `specs` de GCC pour vous assurer que GCC utilise notre environnement de construction :

```
patch -Np1 -i ../gcc-4.6.3-specs-1.patch
```

Modifiez la spec `StartFile` et le `Standard Include Dir` afin que GCC regarde dans `/tools` :

```
echo -en '#undef STANDARD_INCLUDE_DIR\n#define STANDARD_INCLUDE_DIR "/tools/include\n'
echo -en '\n#undef STANDARD_STARTFILE_PREFIX_1\n#define STANDARD_STARTFILE_PREFIX_1 "/tools\n'
echo -en '\n#undef STANDARD_STARTFILE_PREFIX_2\n#define STANDARD_STARTFILE_PREFIX_2 "/tools\n'
```

Maintenant, modifiez le chemin de recherche include par défaut du préprocesseur c de GCC pour n'utiliser que `/tools` :

```
cp -v gcc/Makefile.in{,.orig}
sed -e "s@\(^CROSS_SYSTEM_HEADER_DIR =\) .*@1 /tools/include@g" \
    gcc/Makefile.in.orig > gcc/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 :

```
AR=ar LDFLAGS="-Wl,-rpath,/cross-tools/lib" \
../gcc-4.6.3/configure --prefix=/cross-tools \
--build=${CLFS_HOST} --target=${CLFS_TARGET} --host=${CLFS_HOST} \
--with-sysroot=${CLFS} --with-local-prefix=/tools --disable-nls \
--enable-shared --disable-static --enable-languages=c,c++ \
--enable-__cxa_atexit --with-mpfr=/cross-tools --with-gmp=/cross-tools \
--enable-c99 --with-ppl=/cross-tools --with-cloog=/cross-tools \
--enable-long-long --enable-threads=posix --enable-cloog-backend=isl
```

Voici la signification des nouvelles options de `configure` :

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

Cette option nous assure que seuls les compilateurs C et C++ sont construits.

`--enable-__cxa_atexit`

Cette option permet l'utilisation de `__cxa_atexit`, plutôt que de `atexit`, pour enregistrer les destructeurs C++ pour les statiques locales et les objets globaux, et elle sert essentiellement pour une gestion des destructeurs respectant totalement les standards. Il affecte aussi les ABI C++, ce qui produit des bibliothèques C++ partagées et des programmes C++ interopérables avec d'autres distributions Linux.

--enable-c99

Active le support C99 pour les programmes C.

--enable-long-long

Active le support du type long long dans le compilateur.

--enable-threads=posix

Ceci active la gestion d'exception C++ pour le code multi-tâches.

Continuez en compilant le paquet :

```
make AS_FOR_TARGET="${CLFS_TARGET}-as" \  
LD_FOR_TARGET="${CLFS_TARGET}-ld"
```

Installez le paquet :

```
make install
```

Les détails sur ce paquet sont situés dans Section 10.23.2, « Contenu de GCC. »

Partie IV. Construction des outils de base

Chapitre 6. Construire un système temporaire

6.1. Introduction

Ce chapitre montre comment construire un système Linux minimal. Ce système ne contiendra que les outils nécessaires pour commencer la construction du système CLFS final dans Installing Basic System Software et de créer un environnement de travail avec plus de facilité pour l'utilisateur que ne le permettrait un environnement minimum.

Les outils construits dans ce chapitre sont compilés de manière croisés en utilisant la chaîne d'outils dans `/cross-tools` et seront installés sous le répertoire `${CLFS}/tools` de façon à les garder séparés des fichiers installés dans Installing Basic System Software 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 CLFS.

Vérifiez une dernière fois que la variable d'environnement `CLFS` est correctement paramétrée :

```
echo ${CLFS}
```

Assurez-vous que la sortie montre le chemin vers le point de montage de la partition CLFS qui est `/mnt/clfs`, en utilisant notre exemple.

Pendant cette section de la compilation, vous verrez plusieurs messages d'AVERTISSEMENT (WARNING) comme celui ci-dessous. Vous pouvez ignorer ces messages en toute sécurité.

```
configure: WARNING: If you wanted to set the --build type, don't use --host.
    If a cross compiler is detected then cross compile mode will be used.
```

6.2. Variables de construction

Initialisez les variables spécifiques à la cible pour le compilateur et les éditeurs de liens :

```
export CC="${CLFS_TARGET}-gcc"
export CXX="${CLFS_TARGET}-g++"
export AR="${CLFS_TARGET}-ar"
export AS="${CLFS_TARGET}-as"
export RANLIB="${CLFS_TARGET}-ranlib"
export LD="${CLFS_TARGET}-ld"
export STRIP="${CLFS_TARGET}-strip"
```

Puis ajoutez les variables de construction à `~/ .bashrc` pour éviter les problèmes si vous vous arrêtez et reprenez plus tard :

```
echo export CC=\"\"${CC}\"\" >> ~/.bashrc
echo export CXX=\"\"${CXX}\"\" >> ~/.bashrc
echo export AR=\"\"${AR}\"\" >> ~/.bashrc
echo export AS=\"\"${AS}\"\" >> ~/.bashrc
echo export RANLIB=\"\"${RANLIB}\"\" >> ~/.bashrc
echo export LD=\"\"${LD}\"\" >> ~/.bashrc
echo export STRIP=\"\"${STRIP}\"\" >> ~/.bashrc
```

6.3. GMP-5.0.5

GMP est une bibliothèque pour faire de l'arithmétique en précision arbitraire sur les entiers, les nombres rationnels et les nombres flottants.

6.3.1. Installation de GMP

Préparez la compilation de GMP:

```
HOST_CC=gcc CPPFLAGS=-fexceptions CC="${CC} \  
  ${BUILD64}" CXX="${CXX} ${BUILD64}" ./configure --prefix=/tools \  
  --build=${CLFS_HOST} --host=${CLFS_TARGET} \  
  --libdir=/tools/lib64 --enable-cxx
```

Compilez le paquet :

```
make
```

Installez le paquet :

```
make install
```

Les détails sur ce paquet sont disponibles dans Section 10.11.2, « Contenu de GMP. »

6.4. MPFR-3.1.1

La bibliothèque MPFR est une bibliothèque C pour des calculs de nombres flottants à précision multiple avec un arrondis correct.

6.4.1. Installation de MPFR

Préparez la compilation de MPFR :

```
CC="${CC} ${BUILD64}" ./configure --prefix=/tools \  
  --build=${CLFS_HOST} --host=${CLFS_TARGET} \  
  --libdir=/tools/lib64 --enable-shared
```

Compilez le paquet :

```
make
```

Installez le paquet :

```
make install
```

Les détails sur ce paquet sont situés dans Section 10.13.2, « Contenu de MPFR. »

6.5. MPC-1.0.1

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

6.5.1. Installation de MPC

Préparez la compilation de MPC :

```
CC="${CC} ${BUILD64}" \  
./configure --prefix=/tools \  
--build=${CLFS_HOST} --host=${CLFS_TARGET} \  
--libdir=/tools/lib64
```

Compilez le paquet :

```
make
```

Installez le paquet :

```
make install
```

Les détails sur ce paquet sont disponibles dans Section 10.15.2, « Contenu de MPC. »

6.6. PPL-0.12.1

La bibliothèque *Parma Polyhedra Library* (PPL) fournit des abstractions numériques destinées principalement à des applications dans le domaine de l'analyse et de la vérification de systèmes complexes. CLooG-PPL exige cette bibliothèque.

6.6.1. Installation de PPL

Préparez la compilation de PPL :

```
CC="${CC} ${BUILD64}" ./configure --prefix=/tools \
  --build=${CLFS_HOST} --host=${CLFS_TARGET} \
  --enable-interfaces="c,cxx" --libdir=/tools/lib64 --enable-shared \
  --enable-interfaces="c,cxx" --libdir=/tools/lib64 \
  --enable-shared --disable-optimization \
  --with-gmp-include=/tools/include --with-gmp-lib=/tools/lib
```

Compilez le paquet :

```
make
```

Installez le paquet :

```
make install
```

Les détails sur ce paquet sont situés dans Section 10.17.2, « Contenu de PPL. »

6.7. CLooG-0.16.3

CLooG est une bibliothèque pour générer du code pour analyser des polyèdres Z. En d'autres termes, il trouve du code qui atteint chaque point entier (ou intégral) d'un ou plusieurs polyèdres paramétrés. GCC se lie à cette bibliothèque afin d'activer le nouveau code de génération de boucle, connu en tant que Graphite.

6.7.1. Installation de CLooG

Ce qui suit empêche le script configure de définir LD_LIBRARY_PATH quand il trouve PPL. Cela empêchera tout any conflit avec des bibliothèques du système hôte :

```
cp -v configure{,.orig}
sed -e "/LD_LIBRARY_PATH=/d" \
    configure.orig > configure
```

Préparez la compilation de CLooG :

```
CC="${CC} ${BUILD64}" ./configure --prefix=/tools \
    --build=${CLFS_HOST} --host=${CLFS_TARGET} --libdir=/tools/lib64 \
    --enable-shared --with-gmp-prefix=/tools
```

Compilez le paquet :

```
make
```

Installez le paquet :

```
make install
```

Les dÃ©tails sur ce paquet sont disponibles dans Section 10.19.2, « Contenu de CLooG. »

6.8. Zlib-1.2.7

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

6.8.1. Installation de Zlib

Préparez la compilation de Zlib :

```
CC="${CC} ${BUILD64}" \  
./configure --prefix=/tools --shared --libdir=/tools/lib64
```

Compilez le paquet :

```
make
```

Installez le paquet :

```
make install
```

Les détails sur ce paquet sont disponibles dans Section 10.21.2, « Contenu de Zlib. »

6.9. Binutils-2.23

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

6.9.1. Installation de Binutils

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 :

```
CC="${CC} ${BUILD64}" ../binutils-2.23/configure \
  --prefix=/tools --libdir=/tools/lib64 --with-lib-path=/tools/lib64:/tools/
  --build=${CLFS_HOST} --host=${CLFS_TARGET} --target=${CLFS_TARGET} \
  --disable-nls --enable-shared --enable-64-bit-bfd
```

Voici la signification des options de configure :

```
CC="${CC} ${BUILD64}"
```

Dit au compilateur d'utiliser nos drapeaux 64 bits.

Compilez le paquet :

```
make configure-host
make
```

Installez le paquet :

```
make install
```

Les détails sur ce paquet sont situés dans Section 10.22.2, « Contenu de Binutils. »

6.10. GCC-4.6.3

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

6.10.1. Installation de GCC

Le correctif suivant contient un certain nombre de mises à jour vers la branche 4.6.3, faites par les développeurs de GCC :

```
patch -Np1 -i ../gcc-4.6.3-branch_update-2.patch
```

Faites deux ajustements essentiels pour le fichier specs de GCC pour vous assurer que GCC utilise notre environnement de construction :

```
patch -Np1 -i ../gcc-4.6.3-specs-1.patch
```

Modifiez la spec StartFile et le Standard Include Dir afin que GCC regarde dans `/tools` :

```
echo -en '#undef STANDARD_INCLUDE_DIR\n#define STANDARD_INCLUDE_DIR "/tools/include'\n\n#undef STANDARD_STARTFILE_PREFIX_1\n#define STANDARD_STARTFILE_PREFIX_1 "/tools/startfile'\n\n#undef STANDARD_STARTFILE_PREFIX_2\n#define STANDARD_STARTFILE_PREFIX_2 "/tools/startfile'
```

En outre, nous avons besoin de régler le répertoire recherché par le processus `fixincludes` pour les en-têtes du système, afin qu'il ne regarde pas les en-têtes de l'hôte :

```
cp -v gcc/Makefile.in{,.orig}
sed -e 's@\(^NATIVE_SYSTEM_HEADER_DIR =\) .*@1 /tools/include@g' \
    gcc/Makefile.in.orig > gcc/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
```

Avant de commencer la construction de GCC, souvenez-vous de désinitialiser les variables d'environnement qui surchargent les drapeaux d'optimisation par défaut.

Préparez la compilation de GCC :

```
CC="${CC} ${BUILD64}" CXX="${CXX} ${BUILD64}" \
  ../gcc-4.6.3/configure --prefix=/tools \
  --libdir=/tools/lib64 --build=${CLFS_HOST} --host=${CLFS_TARGET} \
  --target=${CLFS_TARGET} --with-local-prefix=/tools --enable-long-long \
  --enable-c99 --enable-shared --enable-threads=posix \
  --enable-__cxa_atexit --disable-nls --enable-languages=c,c++ \
  --disable-libstdcxx-pch --enable-cloog-backend=isl
```

Voici la signification des nouvelles options de configure :

```
CXX="${CXX} ${BUILD64}"
```

Ceci oblige le compilateur C++ à utiliser nos drapeaux 64 bits.

```
--disable-libstdcxx-pch
```

Ne construit pas l'en-tête précompilée, ou *pre-compiled header* (PCH) pour `libstdc++`. Elle prend beaucoup d'espace et nous n'en avons pas d'utilité pour le moment.

Ce qui suit empêchera GCC de chercher les en-têtes et les bibliothèques dans de mauvais répertoires :

```
cp -v Makefile{,.orig}
sed "/^HOST_\(GMP\|PPL\|CLOOG\)\(LIBS\|INC\) /s:-[IL]/\ (lib\|include\)::" \
    Makefile.orig > Makefile
```

Compilez le paquet :

```
make AS_FOR_TARGET="${AS}" \
    LD_FOR_TARGET="${LD}"
```

Installez le paquet :

```
make install
```

Les détails sur ce paquet sont situés dans Section 10.23.2, « Contenu de GCC. »

6.11. Ncurses-5.9

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

6.11.1. Installation de Ncurses

Le correctif suivant corrige des problèmes avec certaines versions de Bash :

```
patch -Np1 -i ../ncurses-5.9-bash_fix-1.patch
```

Préparez la compilation de Ncurses :

```
CC="${CC} ${BUILD64}" CXX="${CXX} ${BUILD64}" \
./configure --prefix=/tools --with-shared \
--build=${CLFS_HOST} --host=${CLFS_TARGET} \
--without-debug --without-ada \
--enable-overwrite --with-build-cc=gcc \
--libdir=/tools/lib64
```

Voici la signification des nouvelles options de configure :

--with-shared

Ceci dit à Ncurses de créer une bibliothèque partagée.

--without-debug

Ceci dit à Ncurses de ne pas se construire avec les informations de débogage.

--without-ada

Ceci nous assure que Ncurses ne construise pas le support pour le compilateur Ada qui peut être présent sur l'hôte mais qui ne sera pas disponible lors de la construction du système final.

--enable-overwrite

Ceci dit à Ncurses d'installer ses fichiers d'en-tête dans `/tools/include` au lieu de `/tools/include/ncurses`, pour nous assurer que d'autres paquets puissent trouver les en- têtes Ncurses avec succès.

--with-build-cc=gcc

Ceci dit à Ncurses le type de compilateur que nous utilisons.

Compilez le paquet :

```
make
```

Installez le paquet :

```
make install
```

Les dÃ©tails sur ce paquet sont disponibles dans Section 10.27.2, « Contenu de Ncurses. »

6.12. Bash-4.2

Le paquet Bash contient le shell Bourne-Again.

6.12.1. Installation de Bash

Le correctif suivant contient des mises à jour issues du mainteneur. Le mainteneur de Bash ne fait ces correctifs que pour corriger des problèmes sérieux :

```
patch -Np1 -i ../bash-4.2-branch_update-6.patch
```

Quand Bash est compilé de manière croisée, il ne peut notamment pas tester la présence de pipes (tubes) nommés. Si vous avez utilisé **su** pour devenir utilisateur non privilégié, cette combinaison aura pour conséquence que Bash se construira sans *substitution de processus*, ce qui va casser un des scripts de test de C++ dans `eglibc`. Ce qui suit empêche des problèmes futurs en sautant la vérification des tubes nommés et d'autres tests qui ne peuvent pas s'exécuter lors d'une compilation croisée ou qui ne s'exécutent pas correctement :

```
cat > config.cache << "EOF"
ac_cv_func_mmap_fixed_mapped=yes
ac_cv_func_strcoll_works=yes
ac_cv_func_working_mktime=yes
bash_cv_func_sigsetjmp=present
bash_cv_getcwd_malloc=yes
bash_cv_job_control_missing=present
bash_cv_printf_a_format=yes
bash_cv_sys_named_pipes=present
bash_cv_ulimit_maxfds=yes
bash_cv_under_sys_siglist=yes
bash_cv_unusable_rtsigs=no
gt_cv_int_divbyzero_sigfpe=yes
EOF
```

Préparez la compilation de Bash :

```
CC="$CC" CXX="$CXX" \
./configure --prefix=/tools \
--build=${CLFS_HOST} --host=${CLFS_TARGET} \
--without-bash-malloc --cache-file=config.cache
```

Voici la signification de l'option de configure :

`--without-bash-malloc`

Cette option désactive l'utilisation de la fonction d'allocation de mémoire de Bash (`malloc`) qui est connue pour provoquer des erreurs de segmentation. En désactivant cette option, Bash utilisera les fonctions `malloc` de Glibc qui sont plus stables.

Compilez le paquet :

```
make
```

Installez le paquet :

```
make install
```

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

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


Les détails sur ce paquet sont disponibles dans Section 10.52.2, « Contenu de Bash. »

6.13. Bison-2.6.4

Le paquet Bison contient un générateur d'analyseurs.

6.13.1. Installation de Bison

Préparez la compilation de Bison :

```
CC="${CC} ${BUILD64}" ./configure --prefix=/tools \  
--build=${CLFS_HOST} --host=${CLFS_TARGET}
```

Compilez le paquet :

```
make
```

Installez le paquet :

```
make install
```

Les détails sur ce paquet sont situés dans Section 10.40.2, « Contenu de Bison. »

6.14. 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 le classique **gzip**.

6.14.1. Installation de Bzip2

La règle par défaut du Makefile de Bzip2 lance automatiquement la suite de tests. Nous devons supprimer les tests puisqu'ils ne fonctionneront pas sur une construction multi-architectures et nous devons modifier le chemin de la bibliothèque par défaut en `lib64` :

```
cp -v Makefile{,.orig}
sed -e 's@^(all:.*\ ) test@1@g' \
    -e 's@/lib\(/|\ |$\)@/lib64\1@g' Makefile.orig > Makefile
```

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

```
make CC="$CC" ${BUILD64} AR="$AR" RANLIB="$RANLIB"
```

Installez le paquet :

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

Les détails sur ce paquet sont disponibles dans Section 10.54.2, « Contenu de Bzip2. »

6.15. Coreutils-8.20

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

6.15.1. Installation de Coreutils

La commande suivante met à jour les temps indiqués sur les pages de man d'uname et de hostname afin que Makefile ne s'attende pas à les régénérer :

```
touch man/uname.1 man/hostname.1
```

Configure ne peut pas déterminer correctement comment obtenir de l'espace libre lors de la compilation croisée, il en résulte que le programme **df** ne sera pas construit. Ajoutez les entrées suivantes dans `config.cache` pour corriger cela et corrigez divers problèmes de compilation croisée :

```
cat > config.cache << EOF
fu_cv_sys_stat_statfs2_bsize=yes
gl_cv_func_working_mkstemp=yes
EOF
```

Préparez la compilation de Coreutils :

```
CC="$CC" "${BUILD64}" ./configure --prefix=/tools \
  --build=${CLFS_HOST} --host=${CLFS_TARGET} \
  --enable-install-program=hostname --cache-file=config.cache
```

Voici la signification de la nouvelle option de configure :

```
--enable-install-program=hostname
```

Dit à Coreutils d'installer **hostname**, nécessaire à la suite de tests de Perl.

Coreutils ne construit pas `make-prime-list` correctement et il se peut que l'hôte n'exécute pas le binaire cible. Construisez-le en utilisant le compilateur de l'hôte pour qu'on puisse l'exécuter pour la génération des données nécessaires à la construction.

```
cp -v Makefile{,.orig}
sed '/src_make_prime_list/d' Makefile.orig > Makefile
depbased=`echo src/make-prime-list.o | sed 's|[^/]*$|.deps/&;s|\.o$|`';\
gcc -std=gnu99 -I. -I./lib -Ilib -I./lib -Isrc -I./src \
-fdiagnostics-show-option -funit-at-a-time -g -O2 -MT \
src/make-prime-list.o -MD -MP -MF $depbased.Tpo -c -o src/make-prime-list.o \
src/make-prime-list.c &&
mv -f $depbased.Tpo $depbased.Po
gcc -std=gnu99 -fdiagnostics-show-option -funit-at-a-time -g -O2 \
-Wl,--as-needed -o src/make-prime-list src/make-prime-list.o
```

Supprimez la construction de la page de man de hostname car elle est modifiée par les commandes précédentes.

```
cp -v Makefile{,.bak}
sed -e '/hostname.1/d' Makefile.bak > Makefile
```

Compilez le paquet :

```
make
```

Installez le paquet :

```
make install
```

Les détails sur ce paquet sont situés dans Section 10.36.2, « Contenu de Coreutils. »

6.16. Diffutils-3.2

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

6.16.1. Installation de Diffutils

Préparez la compilation de Diffutils :

```
CC="${CC} ${BUILD64}" ./configure --prefix=/tools \  
--build=${CLFS_HOST} --host=${CLFS_TARGET}
```

Compilez le paquet :

```
make
```

Installez le paquet :

```
make install
```

Les détails sur ce paquet sont disponibles dans Section 10.55.2, « Contenu de Diffutils. »

6.17. 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).

6.17.1. Installation de Findutils

Les entrées de cache suivantes règlent les valeurs des tests qui ne se lançaient pas lors de la compilation croisée :

```
echo "gl_cv_func_wcwidth_works=yes" > config.cache
echo "ac_cv_func_fnmatch_gnu=yes" >> config.cache
```

Préparez la compilation de Findutils :

```
CC="${CC} ${BUILD64}" ./configure --prefix=/tools \
  --build=${CLFS_HOST} --host=${CLFS_TARGET} \
  --cache-file=config.cache
```

Compilez le paquet :

```
make
```

Installez le paquet :

```
make install
```

Les détails sur ce paquet sont disponibles dans Section 10.59.2, « Contenu de Findutils. »

6.18. File-5.11

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

6.18.1. Installation de File

Préparez la compilation de File :

```
CC="${CC} ${BUILD64}" ./configure --prefix=/tools \  
--libdir=/tools/lib64 --build=${CLFS_HOST} --host=${CLFS_TARGET}
```

Compilez le paquet :

```
make
```

Installez le paquet :

```
make install
```

Les détails sur ce paquet sont disponibles dans Section 10.57.2, « Contenu de File. »

6.19. Flex-2.5.37

Le paquet Flex contient un outil de génération de programmes reconnaissant des motifs de texte.

6.19.1. Installation de Flex

Lors de la compilation croisée, le script **configure** ne détermine pas la bonne valeur pour ce qui suit. Réglez les valeurs manuellement :

```
cat > config.cache << EOF
ac_cv_func_malloc_0_nonnull=yes
ac_cv_func_realloc_0_nonnull=yes
EOF
```

Préparez la compilation de Flex :

```
CC="${CC} ${BUILD64}" ./configure --prefix=/tools \
  --build=${CLFS_HOST} --host=${CLFS_TARGET} \
  --cache-file=config.cache
```

Compilez le paquet :

```
make
```

Installez le paquet :

```
make install
```

Les détails sur ce paquet sont situés dans Section 10.44.2, « Contenu de Flex. »

6.20. Gawk-4.0.1

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

6.20.1. Installation de Gawk

Préparez la compilation de Gawk :

```
CC="${CC} ${BUILD64}" ./configure --prefix=/tools \  
--build=${CLFS_HOST} --host=${CLFS_TARGET}
```

Compilez le paquet :

```
make
```

Installez le paquet :

```
make install
```

Les détails sur ce paquet sont situés dans Section 10.58.2, « Contenu de Gawk. »

6.21. Gettext-0.18.1.1

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.

6.21.1. Installation de Gettext

Seuls les programmes du répertoire `gettext-tools` doivent être installés dans le système temporaire :

```
cd gettext-tools
```

Lors d'une compilation croisée, le script configure de Gettext suppose que nous n'avons pas de `wcwidth` fonctionnel alors que c'est le cas. Ce qui suit va corriger des erreurs de compilation possibles dues à ces présupposés :

```
echo "gl_cv_func_wcwidth_works=yes" > config.cache
```

Préparez la compilation de Gettext :

```
CC="${CC} ${BUILD64}" CXX="${CXX} ${BUILD64}" \
./configure --prefix=/tools --disable-shared \
--build=${CLFS_HOST} --host=${CLFS_TARGET} \
--cache-file=config.cache
```

Voici la signification des options de configure :

`--disable-shared`

Ceci dit à Gettext de ne pas créer de bibliothèque partagée.

Compilez le paquet :

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

Installez le binaire `msgfmt` :

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

Les détails sur ce paquet sont disponibles dans Section 10.61.2, « Contenu de Gettext. »

6.22. Grep-2.14

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

6.22.1. Installation de Grep

En compilation croisée, le script **configure** ne détermine pas les bonnes valeurs pour ce qui suit. Paramétrez les valeurs à la main :

```
cat > config.cache << EOF
ac_cv_func_malloc_0_nonnull=yes
ac_cv_func_realloc_0_nonnull=yes
EOF
```

Préparez la compilation de Grep :

```
CC="${CC} ${BUILD64}" ./configure --prefix=/tools \
  --build=${CLFS_HOST} --host=${CLFS_TARGET} \
  --without-included-regex --cache-file=config.cache
```

Voici la signification de la nouvelle option de configure :

--without-included-regex

Lors d'une compilation croisée, le script **configure** de Grep suppose qu'il n'y a aucune installation non utilisable de `regex.h` et il utilise à la place celui inclu dans Grep. Ce paramètre oblige à utiliser les fonctions regex d'EGLIBC.

Compilez le paquet :

```
make
```

Installez le paquet :

```
make install
```

Les détails sur ce paquet sont disponibles dans Section 10.62.2, « Contenu de Grep. »

6.23. Gzip-1.5

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

6.23.1. Installation de Gzip

Préparez la compilation de Gzip :

```
CC="${CC} ${BUILD64}" ./configure --prefix=/tools \  
--build=${CLFS_HOST} --host=${CLFS_TARGET}
```

Compilez le paquet :

```
make
```

Installez le paquet :

```
make install
```

Les détails sur ce paquet sont disponibles dans Section 10.65.2, « Contenu de Gzip. »

6.24. M4-1.4.16

Le paquet M4 contient un processeur de macros.

6.24.1. Installation de M4

Configure ne peut pas déterminer correctement les résultats des tests suivants :

```
cat > config.cache << EOF
gl_cv_func_btowc_eof=yes
gl_cv_func_mbrtowc_incomplete_state=yes
gl_cv_func_mbrtowc_sanitycheck=yes
gl_cv_func_mbrtowc_null_arg=yes
gl_cv_func_mbrtowc_retval=yes
gl_cv_func_mbrtowc_nul_retval=yes
gl_cv_func_wcrtomb_retval=yes
gl_cv_func_wctob_works=yes
EOF
```

Préparez la compilation de M4 :

```
CC="${CC} ${BUILD64}" ./configure --prefix=/tools \
  --build=${CLFS_HOST} --host=${CLFS_TARGET} \
  --cache-file=config.cache
```

Compilez le paquet :

```
make
```

Installez le paquet :

```
make install
```

Les détails sur ce paquet sont situés dans Section 10.38.2, « Contenu de M4. »

6.25. Make-3.82

Le paquet Make contient un programme pour compiler des paquets.

6.25.1. Installation de Make

Préparez la compilation de Make :

```
CC="${CC} ${BUILD64}" ./configure --prefix=/tools \  
--build=${CLFS_HOST} --host=${CLFS_TARGET}
```

Compilez le paquet :

```
make
```

Installez le paquet :

```
make install
```

Les détails sur ce paquet sont disponibles dans Section 10.68.2, « Contenu de Make. »

6.26. 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**.

6.26.1. Installation de Patch

Préparez la compilation de Patch :

Quand le configure à la compilation croisée ne peut pas détecter la présence de certaines fonctionnalités, modifiez ce comportement :

```
echo "ac_cv_func_strnlen_working=yes" > config.cache
```

```
CC="${CC} ${BUILD64}" ./configure --prefix=/tools \  
  --build=${CLFS_HOST} --host=${CLFS_TARGET} \  
  --cache-file=config.cache
```

Compilez le paquet :

```
make
```

Installez le paquet :

```
make install
```

Les détails sur ce paquet sont disponibles dans Section 10.74.2, « Contenu de Patch. »

6.27. Sed-4.2.1

Le paquet Sed contient un éditeur de flux.

6.27.1. Installation de Sed

Préparez la compilation de Sed :

```
CC="${CC} ${BUILD64}" ./configure --prefix=/tools \  
--build=${CLFS_HOST} --host=${CLFS_TARGET}
```

Compilez le paquet :

```
make
```

Installez le paquet :

```
make install
```

Les détails sur ce paquet sont disponibles dans Section 10.25.2, « Contenu de Sed. »

6.28. Tar-1.26

Le paquet Tar contient un programme d'archivage.

6.28.1. Installation de Tar

Configure ne peut déterminer le résultat de quelques tests. Réglez-les manuellement :

```
cat > config.cache << EOF
gl_cv_func_wcwidth_works=yes
gl_cv_func_btowc_eof=yes
ac_cv_func_malloc_0_nonnull=yes
ac_cv_func_realloc_0_nonnull=yes
gl_cv_func_mbrtowc_incomplete_state=yes
gl_cv_func_mbrtowc_nul_retval=yes
gl_cv_func_mbrtowc_null_arg=yes
gl_cv_func_mbrtowc_retval=yes
gl_cv_func_wcrtomb_retval=yes
EOF
```

Préparez la compilation de Tar :

```
CC="${CC} ${BUILD64}" ./configure --prefix=/tools \
  --build=${CLFS_HOST} --host=${CLFS_TARGET} \
  --cache-file=config.cache
```

Compilez le paquet :

```
make
```

Installez le paquet :

```
make install
```

Les détails sur ce paquet sont disponibles dans Section 10.82.2, « Contenu de Tar. »

6.29. Texinfo-4.13a

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

6.29.1. Installation de Texinfo

Préparez la compilation de Texinfo :

```
CC="${CC} ${BUILD64}" ./configure --prefix=/tools \  
--build=${CLFS_HOST} --host=${CLFS_TARGET}
```

Compilez le paquet :

```
make -C tools/gnulib/lib  
make -C tools  
make
```

Installez le paquet :

```
make install
```

Les détails sur ce paquet sont disponibles dans Section 10.83.2, « Contenu de Texinfo. »

6.30. Vim-7.3

Le paquet Vim contient un puissant éditeur de texte.

6.30.1. Installation de VIM

Le script configure a un seul test en dur qui ne peut pas réussir par une entrée de cache. Désactivez ce test avec la commande suivante :

```
cp -v src/auto/configure{,.orig}
sed "/using uint32_t/s/as_fn_error/#&/" src/auto/configure.orig > src/auto/con
```

Le correctif suivant incorpore toutes les mises à jour de la branche 7.3 issue des développeurs de Vim :

```
patch -Np1 -i ../vim-7.3-branch_update-6.patch
```

Le script **configure** est tel qu'il s'arrête au premier signe d'une compilation croisée. Améliorez cela en initialisant les valeurs en cache de ces tests avec la commande suivante :

```
cat > src/auto/config.cache << "EOF"
vim_cv_getcwd_broken=no
vim_cv_memmove_handles_overlap=yes
vim_cv_stat_ignores_slash=no
vim_cv_terminfo=yes
vim_cv_tgent=zero
vim_cv_toupper_broken=no
vim_cv_tty_group=world
ac_cv_sizeof_int=4
ac_cv_sizeof_long=4
ac_cv_sizeof_time_t=4
ac_cv_sizeof_off_t=4
EOF
```

Modifiez l'emplacement par défaut du fichier de configuration vimrc vers /tools/etc :

```
echo '#define SYS_VIMRC_FILE "/tools/etc/vimrc"' >> src/feature.h
```

Préparez la compilation de Vim :

```
CC="${CC} ${BUILD64}" CXX="${CXX} ${BUILD64}" \
./configure --build=${CLFS_HOST} --host=${CLFS_TARGET} \
--prefix=/tools --enable-multibyte --enable-gui=no \
--disable-gtktest --disable-xim --with-features=normal \
--disable-gpm --without-x --disable-netbeans \
--with-tlib=ncurses
```

Compilez le paquet :

```
make
```

Installez le paquet :

```
make install
```

Beaucoup d'utilisateurs sont habitués à utiliser **vi** au lieu de **vim**. Certains programmes comme **vigr** et **vipw** utilisent aussi **vi**. Créez un lien symbolique pour permettre l'exécution de **vim** lorsque les utilisateurs entrent habituellement **vi** et pour permettre aux programmes qui utilisent **vi** de fonctionner :

```
ln -sv vim /tools/bin/vi
```

Créez un vimrc temporaire pour qu'il fonctionne davantage selon la manière à laquelle vous pourriez vous attendre. C'est expliqué plus amplement dans le système final :

```
cat > /tools/etc/vimrc << "EOF"
" Début de /etc/vimrc

set nocompatible
set backspace=2
set ruler
syntax on

" Fin de /etc/vimrc
EOF
```

Les détails sur ce paquet sont situés dans Section 10.86.3, « Contenu de Vim. »

6.31. XZ Utils-5.0.4

Le paquet XZ-Utils contient des programmes pour compresser et décompresser des fichiers. La compression de fichiers texte avec **XZ-Utils** donne un pourcentage de compression bien meilleur qu'avec le **gzip** traditionnel.

6.31.1. Installation de XZ-Utils

Préparez la compilation de XZ-Utils :

```
CC="${CC} ${BUILD64}" ./configure --prefix=/tools \  
  --build=${CLFS_HOST} --host=${CLFS_TARGET} \  
  --libdir=/tools/lib64
```

Compilez le paquet :

```
make
```

Installez le paquet :

```
make install
```

Les détails sur ce paquet sont situés dans Section 10.70.2, « Contenu de XZ-Utils. »

6.32. Démarrer ou se chrooter ?

Il y a deux principales manières de poursuivre à partir de ce moment pour construire le système final. Vous pouvez construire un noyau, un chargeur de démarrage et quelques autres outils, démarrer dans le système temporaire et y construire le reste. Vous pouvez également vous chrooter dans le système temporaire.

La méthode de démarrage est nécessaire quand vous construisez sur une architecture différente. Par exemple, si vous construisez un système PowerPC à partir d'un x86, vous ne pouvez pas vous chrooter. La méthode chroot vaut quand vous construisez sur la même architecture. Si vous construisez sur et pour un système x86, vous pouvez simplement vous chrooter. La règle d'or ici est que si les architectures correspondent et que vous exécutez la même série du noyau, vous pouvez simplement vous chrooter. Si vous n'exécutez pas sur une même série de noyau, ou si vous voulez exécuter un ABI différente, vous aurez besoin d'utiliser les options de démarrage.

Si vous avez un doute à ce sujet, vous pouvez essayer les commandes suivantes pour voir si vous pouvez chroot :

```
/tools/lib/libc.so.6  
/tools/lib64/libc.so.6  
/tools/bin/gcc -v
```

Si l'une de ces commandes échoue, vous devrez choisir la méthode du démarrage.

Pour vous chrooter, vous aurez également besoin d'un noyau Linux 2.6.32 ou supérieur (compilé avec GCC-4.1.2 ou supérieur). La raison expliquant cette exigence de la version du noyau est qu'eglbc est construit pour générer une bibliothèque pour la version du noyau Linux qui devrait être supportée la plus petite.

Pour vérifier la version de votre noyau, lancez **cat /proc/version** - si elle ne dit pas que vous exécutez un noyau Linux 2.6.32 ou supérieur compilé avec GCC 4.1.2 ou supérieur, vous ne pouvez pas vous chrooter.

Pour la méthode de démarrage, suivez le [If You Are Going to Boot](#).

Pour la méthode chroot, suivez le [If You Are Going to Chroot](#).

Chapitre 7. Si vous allez redémarrer

7.1. Introduction

Ce chapitre montre comment compléter la construction des outils temporaire pour créer un système minimal qui sera utilisé pour démarrer la machine cible et pour construire les paquets du système final.

Il y a quelques paquets supplémentaires à installer pour vous permettre de démarrer le système minimal. Certains de ces paquets seront installés à la racine ou dans `/usr` sur la partition CLFS (`${CLFS}/bin`, `${CLFS}/usr/bin`, ...), et non dans `/tools`, en utilisant l'option "DESTDIR" avec `make`. Ceci imposera que l'utilisateur `clfs` ait les droits d'écriture sur le reste de la partition CLFS, donc vous aurez besoin de modifier temporairement le propriétaire de `${CLFS}` pour qu'il appartienne à l'utilisateur `clfs`. Lancez la commande suivante en tant que `root` :

```
chown -v clfs ${CLFS}
```

7.2. Créer les répertoires

Il est temps de créer une structure sur le système de fichiers CLFS. Créez une arborescence de répertoires standard en lançant les commandes suivantes :

```
mkdir -pv ${CLFS}/{bin,boot,dev,{etc/,}opt,home,lib{,64},mnt}
mkdir -pv ${CLFS}/{proc,media/{floppy,cdrom},run/{,shm},sbin,srv,sys}
mkdir -pv ${CLFS}/var/{lock,log,mail,spool}
mkdir -pv ${CLFS}/var/{opt,cache,lib{,64}/{misc,locate},local}
install -dv ${CLFS}/root -m 0750
install -dv ${CLFS}/{/var,}/tmp -m 1777
mkdir -pv ${CLFS}/usr/{,local}/{bin,include,lib{,64},sbin,src}
mkdir -pv ${CLFS}/usr/{,local}/share/{doc,info,locale,man}
mkdir -pv ${CLFS}/usr/{,local}/share/{misc,terminfo,zoneinfo}
mkdir -pv ${CLFS}/usr/{,local}/share/man/man{1,2,3,4,5,6,7,8}
for dir in ${CLFS}/usr{,/local}; do
    ln -sv share/{man,doc,info} $dir
done
install -dv ${CLFS}/usr/lib/locale
ln -sv ../lib/locale ${CLFS}/usr/lib64
```

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 ont été effectuées : une pour le répertoire principal de `root`, et une autre pour les répertoires des fichiers temporaires.

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

7.2.1. 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 l'arborescence créée ci-dessus, 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.

7.3. Création des liens essentiels

Certains programmes utilisent des chemins liés en dur à des programmes qui n'existent pas encore. Afin de satisfaire ces programmes, créez un certain nombre de liens symboliques qui seront remplacés par des fichiers réels tout au long du chapitre suivant après que le logiciel a été installé.

```
ln -sv /tools/bin/{bash,cat,echo,grep,login,passwd,pwd,sleep,stty} ${CLFS}/bin
ln -sv /tools/bin/file ${CLFS}/usr/bin
ln -sv /tools/sbin/{agetty,blkid} ${CLFS}/sbin
ln -sv /tools/lib/libgcc_s.so{,.1} ${CLFS}/usr/lib
ln -sv /tools/lib64/libgcc_s.so{,.1} ${CLFS}/usr/lib64
ln -sv /tools/lib/libstd*so* ${CLFS}/usr/lib
ln -sv /tools/lib64/libstd*so* ${CLFS}/usr/lib64
ln -sv bash ${CLFS}/bin/sh
ln -sv ../run ${CLFS}/var/run
```

7.4. Util-linux-2.22.1

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.

7.4.1. Installation de Util-linux

Préparez la compilation d'Util-linux :

```
CC="${CC} ${BUILD64}" PKG_CONFIG=true ./configure \
  --prefix=/tools --exec-prefix="" --build=${CLFS_HOST} \
  --host=${CLFS_TARGET} --libdir=/tools/lib64 --bindir=/tools/bin \
  --sbindir=/tools/sbin --disable-makeinstall-chown --disable-login \
  --disable-su --config-cache
```

Compilez le paquet :

```
make
```

Installez le paquet :

```
make usrsbin_execdir=/tools/sbin usrbin_execdir=/tools/bin install
```

Les détails sur ce paquet sont disponibles dans Section 10.30.3, « Contenu de Util-linux. »

7.5. Shadow-4.1.5.1

Le paquet Shadow contient des programmes de gestion de mots de passe d'une façon sécurisée.

7.5.1. Installation de Shadow

Désactivez l'installation du programme **groups**, car Coreutils offre une meilleure version :

```
cp -v src/Makefile.in{,.orig}
sed -e 's/groups$(EXEEXT) //' src/Makefile.in.orig > src/Makefile.in
```

Les entrées de cache suivantes définissent les valeurs des tests qui ne se lancent pas dans une compilation croisée :

```
echo "ac_cv_func_setpgrp_void=yes" > config.cache
```

Préparez la compilation de Shadow :

```
CC="${CC} ${BUILD64}" ./configure --prefix=/tools \
  --build=${CLFS_HOST} --host=${CLFS_TARGET} --sysconfdir=/etc \
  --cache-file=config.cache
```

Voici la signification des options de configure :

--sysconfdir=/etc

Dit à Shadow d'installer ses fichiers de configuration dans */etc* et non dans */tools/etc*.

Compilez le paquet :

```
make
```

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

Installez le paquet :

```
make DESTDIR=${CLFS} install
```

Les détails sur ce paquet sont situés dans Section 10.35.4, « Contenu de Shadow. »

7.6. E2fsprogs-1.42.6

Le paquet E2fsprogs contient les outils de gestion du système de fichiers ext2. Il supporte aussi les systèmes de fichiers journalisés ext3 et ext4.

7.6.1. Installation de E2fsprogs

Assurez-vous que les bibliothèques sont installées dans `/tools/lib64` :

```
cp -v configure{, .orig}
sed -e "/libdir=.*\lib/s@/lib@/lib64@g" configure.orig > configure
```

La documentation d'E2fsprogs recommande de construire le paquet dans un sous-répertoire du répertoire source :

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

Préparez la compilation d'E2fsprogs :

```
CC="${CC} ${BUILD64}" PKG_CONFIG=true \
  ../configure --prefix=/tools --enable-elf-shlibs \
  --build=${CLFS_HOST} --host=${CLFS_TARGET} \
  --disable-libblkid --disable-libuuid --disable-fsck \
  --disable-uuid
```

Voici la signification des options de configure :

`--enable-elf-shlibs`

Ceci crée les bibliothèques partagées que certains programmes de ce paquet utilisent.

Compilez le paquet :

```
make LIBUUID="-luuid" STATIC_LIBUUID="-luuid" \
  LIBBLKID="-liblkid" STATIC_LIBBLKID="-liblkid" \
  LDFLAGS="-Wl,-rpath,/tools/lib64"
```

Installez les binaires, la documentation et les bibliothèques partagées :

```
make install
```

Installez les bibliothèques statiques et les en-têtes :

```
make install-libs
```

Créez des liens symboliques nécessaires pour un système amorçable :

```
ln -sv /tools/sbin/{fsck.ext2,fsck.ext3,fsck.ext4,e2fsck} ${CLFS}/sbin
```

Les détails sur ce paquet sont situés dans Section 10.34.2, « Contenu de E2fsprogs. »

7.7. Sysvinit-2.88dsf

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

7.7.1. Installation de Sysvinit

Les modifications suivantes aident à localiser des fichiers spécifiques à cette construction en particulier :

```
cp -v src/Makefile{,.orig}
sed -e 's,/usr/lib,/tools/lib,g' \
    src/Makefile.orig > src/Makefile
```

Compilez le paquet :

```
make -C src clobber
make -C src CC="${CC} ${BUILD64}"
```

Installez le paquet :

```
make -C src ROOT=${CLFS} install
```

7.7.2. Configurer Sysvinit

Créez un nouveau `${CLFS}/etc/inittab` en exécutant ce qui suit :

```
cat > ${CLFS}/etc/inittab << "EOF"
# Début de /etc/inittab

id:3:initdefault:

si::sysinit:/etc/rc.d/init.d/rc sysinit

10:0:wait:/etc/rc.d/init.d/rc 0
11:S1:wait:/etc/rc.d/init.d/rc 1
12:2:wait:/etc/rc.d/init.d/rc 2
13:3:wait:/etc/rc.d/init.d/rc 3
14:4:wait:/etc/rc.d/init.d/rc 4
15:5:wait:/etc/rc.d/init.d/rc 5
16:6:wait:/etc/rc.d/init.d/rc 6

ca:12345:ctrlaltdel:/sbin/shutdown -t1 -a -r now

su:S016:once:/sbin/sulogin

EOF
```

La commande suivante ajoute les terminaux virtuels standards à `${CLFS}/etc/inittab`. Si votre système n'a qu'une console série, sautez la commande suivante :

```
cat >> ${CLFS}/etc/inittab << "EOF"
1:2345:respawn:/sbin/agetty -I '\033(K' tty1 9600
2:2345:respawn:/sbin/agetty -I '\033(K' tty2 9600
3:2345:respawn:/sbin/agetty -I '\033(K' tty3 9600
4:2345:respawn:/sbin/agetty -I '\033(K' tty4 9600
5:2345:respawn:/sbin/agetty -I '\033(K' tty5 9600
6:2345:respawn:/sbin/agetty -I '\033(K' tty6 9600

EOF
```

Si votre système a une console en série, lancez la commande suivante pour ajouter l'entrée à `${CLFS}/etc/inittab`.

```
cat >> ${CLFS}/etc/inittab << "EOF"
c0:12345:respawn:/sbin/agetty 115200 ttyS0 vt100

EOF
```

Enfin, ajoutez une fin de ligne à `${CLFS}/etc/inittab`.

```
cat >> ${CLFS}/etc/inittab << "EOF"
# End /etc/inittab
EOF
```

Les détails sur ce paquet sont situés dans Section 10.81.3, « Contenu de Sysvinit. »

7.8. Kmod-10

Le paquet Kmod contient des programmes pour charger, insérer et supprimer des modules du noyau pour Linux. Kmod remplace le paquet Module-Init-tools.

7.8.1. Installation de Kmod

Préparez la compilation de Kmod :

```
CC="${CC} ${BUILD64}" ./configure --prefix=/tools \
  --bindir=/bin --build=${CLFS_HOST} --host=${CLFS_TARGET} \
  --libdir=/tools/lib64
```

Compile the package:

```
make
```

Installez le paquet :

```
make DESTDIR=${CLFS} install
```

Créez des liens symboliques pour les programmes qui cherchent Module-Init-Tools.

```
ln -sv kmod ${CLFS}/bin/lsmmod
ln -sv ../bin/kmod ${CLFS}/sbin/depmod
ln -sv ../bin/kmod ${CLFS}/sbin/insmod
ln -sv ../bin/kmod ${CLFS}/sbin/modprobe
ln -sv ../bin/kmod ${CLFS}/sbin/modinfo
ln -sv ../bin/kmod ${CLFS}/sbin/rmmod
```

Les détails sur ce paquet sont disponibles dans Section 10.73.2, « Contenu de Kmod. »

7.9. Udev-182

Le paquet Udev contient des programmes pour créer dynamiquement des nœuds périphériques.

7.9.1. Installation de Udev

Préparez la compilation d'Udev :

```
CC="${CC} ${BUILD64}" LIBS="-lpthread" \
  BLKID_CFLAGS="-I/tools/include/blkid" BLKID_LIBS="-L/tools/lib64 -lblkid" \
  KMOD_CFLAGS="-I/tools/include" KMOD_LIBS="-L${CLFS}/lib64 -lkmod" \
  ./configure --prefix=/usr --build=${CLFS_HOST} --host=${CLFS_TARGET} \
  --with-rootprefix='' --bindir=/sbin --sysconfdir=/etc --libexecdir=/lib \
  --libdir=/usr/lib64 --disable-introspection \
  --with-usb-ids-path=no --with-pci-ids-path=no \
  --disable-gtk-doc-html --disable-gudev --disable-keymap --disable-logging \
  --with-firmware-path=/lib/firmware
```

Compilez le paquet :

```
make
```

Installez le paquet :

```
make DESTDIR=${CLFS} install
```

Les détails sur ce paquet sont situés dans Section 10.85.2, « Contenu de Udev. »

7.10. Créer les fichiers de mot de passe, des groupes et des journaux

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

Créez le fichier `${CLFS}/etc/passwd` en lançant la commande suivante :

```
cat > ${CLFS}/etc/passwd << "EOF"
root::0:0:root:/root:/bin/bash
EOF
```

Le mot de passe pour `root` (le « `::` » utilisé ici n'est qu'un paramètre fictif et vous permet de vous connecter sans mot de passe) sera défini plus tard.

Utilisateurs supplémentaires que vous pourriez vouloir ajouter :

```
bin:x:1:1:bin:/bin:/bin/false
```

Peut être utile pour la compatibilité avec des applications héritées.

```
daemon:x:2:6:daemon:/sbin:/bin/false
```

Il est souvent recommandé d'utiliser l'ID d'un groupe ou d'un utilisateur non privilégiés pour l'exécution de démons, afin de limiter leur accès au système.

```
adm:x:3:16:adm:/var/adm:/bin/false
```

Était utilisé pour des programmes qui effectuaient des tâches d'administration.

```
lp:x:10:9:lp:/var/spool/lp:/bin/false
```

Utilisé par des programmes pour l'impression

```
mail:x:30:30:mail:/var/mail:/bin/false
```

Souvent utilisé par des programmes de messagerie

```
news:x:31:31:news:/var/spool/news:/bin/false
```

Souvent utilisé pour un réseau de serveurs de nouvelles (*news*)

```
operator:x:50:0:operator:/root:/bin/bash
```

Souvent utilisé pour permettre aux opérateurs du système d'accéder au système

```
postmaster:x:51:30:postmaster:/var/spool/mail:/bin/false
```

Utilisé généralement comme compte qui reçoit toutes les informations de problèmes avec le serveur de messagerie

```
nobody:x:65534:65534:nobody:/:/bin/false
```

Utilisé par NFS

Créez le fichier `${CLFS}/etc/group` en lançant la commande suivante :

```
cat > ${CLFS}/etc/group << "EOF"
root:x:0:
bin:x:1:
sys:x:2:
kmem:x:3:
tty:x:4:
tape: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:
EOF
```

Groupes supplémentaires que vous pourriez vouloir ajouter

`adm:x:16:root,adm,daemon`

Tous les utilisateurs de ce groupe ont le droit de faire des tâches d'administration

`console:x:17:`

Ce groupe a un accès direct à la console

`cdrw:x:18:`

Ce groupe est autorisé à utiliser le lecteur CDRW

`mail:x:30:mail`

Utilisé par MTAs (Mail Transport Agents)

`news:x:31:news`

Utilisé par le réseau de serveurs de nouvelles

`users:x:1000:`

Le GID utilisé par défaut par shadow pour les nouveaux utilisateurs

`nogroup:x:65533:`

C'est le groupe par défaut utilisé par certains programmes qui n'ont pas besoin d'un groupe

`nobody:x:65534:`

C'est utilisé par NFS

Les groupes créés ne font partie d'aucun standard—ce sont des groupes décidés d'une part par les exigences de la configuration d'Udev dans le système final, d'autre part par la convention couramment utilisée par un grand nombre de distributions Linux existantes. La *Linux Standard Base* (LSB, disponible sur <http://www.linuxbase.org>) recommande uniquement que, après le groupe « root » ayant l'identifiant de groupe (GID) 0, un groupe « bin » avec un GID de 1 soit présent. L'administrateur système peut choisir librement tout autre noms de groupe et GIDs, vu que les programmes bien écrits ne dépendent pas des numéros GID mais utilisent plutôt le nom d'un groupe.

Les programmes **login**, **agetty** et **init** (et d'autres) utilisent un certain nombre de fichiers journal pour enregistrer des informations telles que ceux qui se sont connectés au système et quand. Néanmoins, ces programmes n'écritont pas dans les fichiers journal s'ils n'existent pas déjà. Initialisez les fichiers journal et donnez-leur les bons droits :

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

Le fichier `/var/run/utmp` enregistre les utilisateurs actuellement connectés. Le fichier `/var/log/wtmp` enregistre toutes les connexions et les déconnexions. Le fichier `/var/log/lastlog` enregistre le moment où chaque utilisateur s'est connecté pour la dernière fois. Le fichier `/var/log/btmp` enregistre les tentatives de connexion erronées.

7.11. Linux-3.4.17

Le paquet Linux contient le noyau Linux.

7.11.1. Installation du noyau



Avertissement

Un noyau temporaire compilé de façon croisée sera ici construit. Lors de sa configuration, sélectionnez un jeu d'options minimal requis pour démarrer la machine cible et construire le système final. Ainsi, aucun support pour le son, les imprimantes, etc ne sera nécessaire.

Essayez d'éviter aussi si possible l'utilisation de modules et n'utilisez pas l'image du noyau finale pour la production de systèmes.

La construction du noyau implique quelques étapes — la configuration, la compilation et l'installation. Lisez le fichier README dans l'arborescence des sources du noyau pour des méthodes alternatives de à celle utilisée par le livre pour configurer le noyau.

To ensure that your system boots and you can properly run both 32 bit and 64 bit binaries, please make sure that you enable support for ELF and emulations for 32bit ELF into the kernel.

Préparez la compilation en lançant la commande suivante :

```
make mrproper
```

Ceci garantit que l'arborescence du noyau est absolument propre. L'équipe du noyau recommande que cette commande soit exécutée avant chaque compilation du noyau. Ne pensez pas que l'arborescence des sources est propre après la décompression.

Configurez le noyau avec l'interface du menu :

```
make ARCH=x86_64 CROSS_COMPILE=${CLFS_TARGET}- menuconfig
```

Compilez l'image et les modules du noyau :

```
make ARCH=x86_64 CROSS_COMPILE=${CLFS_TARGET}-
```

Si vous ne pouvez pas vous passer des modules du noyau, vous pouvez avoir besoin d'un fichier `/etc/modprobe.conf`. Vous trouverez des informations concernant les modules et la configuration du noyau dans la documentation du noyau dans le répertoire `Documentation` de l'arborescence des sources du noyau. La page de man `modprobe.conf` peut aussi être intéressante.

Be very careful when reading other documentation relating to kernel modules because it usually applies to 2.4.x kernels only. As far as we know, kernel configuration issues specific to Hotplug and Udev are not documented. The problem is that Udev will create a device node only if Hotplug or a user-written script inserts the corresponding module into the kernel, and not all modules are detectable by Hotplug. Note that statements like the one below in the `/etc/modprobe.conf` file do not work with Udev:

```
alias char-major-XXX some-module
```

Install the modules, if the kernel configuration uses them:

```
make ARCH=x86_64 CROSS_COMPILE=${CLFS_TARGET}- \  
INSTALL_MOD_PATH=${CLFS} modules_install
```

After kernel compilation is complete, additional steps are required to complete the installation. Some files need to be copied to the `${CLFS}/boot` directory.

Issue the following command to install the kernel:

```
cp -v arch/x86_64/boot/bzImage ${CLFS}/boot/vmlinuz-clfs-3.4.17
```

`System.map` is a symbol file for the kernel. It maps the function entry points of every function in the kernel API, as well as the addresses of the kernel data structures for the running kernel. Issue the following command to install the map file:

```
cp -v System.map ${CLFS}/boot/System.map-3.4.17
```

The kernel configuration file `.config` produced by the **make menuconfig** step above contains all the configuration selections for the kernel that was just compiled. It is a good idea to keep this file for future reference:

```
cp -v .config ${CLFS}/boot/config-3.4.17
```

Les détails sur ce paquet sont situés dans Section 13.3.2, « Contents of Linux. »

7.12. GRUB-2.00

Le paquet GRUB contient le *GRand Unified Bootloader*.

7.12.1. Installation de GRUB



Remarque

Si vous aimeriez utiliser un autre chargeur de démarrage, vous pouvez vous rendre à l'adresse suivante pour des chargeurs de démarrage alternatifs et les instructions pour les utiliser. <http://trac.cross-lfs.org/wiki/bootloaders>

Préparez la construction de GRUB :

```
CC="${CC} ${BUILD64}" ./configure --prefix=/tools \
  --build=${CLFS_HOST} --host=${CLFS_TARGET} \
  --sysconfdir=/etc --libdir=/tools/lib64 --disable-werror
```

Compilez le paquet :

```
make
```

Installez le paquet :

```
make DESTDIR=${CLFS} install
```

Les détails sur ce paquet sont situés dans Section 10.87.3, « Contenu de GRUB. »

7.13. Configurer l'environnement

La nouvelle session du shell qui va commencer lorsque l'on va démarrer le système est un shell de *connexion* qui va lire le fichier `.bash_profile`. Créez maintenant le fichier `.bash_profile` :

```
cat > ${CLFS}/root/.bash_profile << "EOF"
set +h
PS1='\u:\w\$ '
LC_ALL=POSIX
PATH=/bin:/usr/bin:/sbin:/usr/sbin:/tools/bin:/tools/sbin
export LC_ALL PATH PS1
EOF
```

La variable `LC_ALL` contrôle la localisation de certains programmes, en faisant en sorte que leurs messages suivent les conventions d'un pays spécifié. Configurer `LC_ALL` à « POSIX » ou « C » (les deux sont équivalents) assure que tout fonctionnera comme prévu sur votre système temporaire.

En mettant `/tools/bin` et `/tools/sbin` à la fin du `PATH` standard, tous les programmes installés dans le Constructing a Temporary System ne sont pris en compte que par le shell s'ils n'ont pas encore été construits sur le système cible. Cette configuration oblige l'utilisation des binaires du système final tels que construits à partir du système temporaire, ce qui minimise les chances que les programmes du système final soient construits contre le système temporaire.

7.14. Options de construction

Nous allons avoir besoin de copier nos variables de construction vers notre nouveau système :

```
cat >> ${CLFS}/root/.bash_profile << EOF
export BUILD32="${BUILD32}"
export BUILD64="${BUILD64}"
export CLFS_TARGET32="${CLFS_TARGET32}"
EOF
```

7.15. Créer le fichier `/etc/fstab`

Le fichier `/etc/fstab` est utilisé par certains programmes pour déterminer où vont être montés les systèmes de fichiers par défaut, ceux qui doivent être vérifiés et dans quel ordre. Créez une nouvelle table de systèmes de fichiers comme ceci :

```
cat > ${CLFS}/etc/fstab << "EOF"
# Début de /etc/fstab

# Système de fichiers Point de montage Type Options dump fsck
# order

/dev/[xxx] / [fff] defaults 1 1
/dev/[yyy] swap swap pri=1 0 0
proc /proc proc defaults 0 0
sysfs /sys sysfs defaults 0 0
devpts /dev/pts devpts gid=4,mode=620 0 0
shm /dev/shm tmpfs defaults 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 adaptées à votre système, par exemple `hda2`, `hda5` et `ext2`. Pour des détails sur les six champs de ce fichier, voir **man 5 fstab**.

Le point de montage `/dev/shm` pour `tmpfs` est inclu pour permettre l'activation de la mémoire partagée POSIX. Le noyau doit disposer du support requis en interne pour fonctionner (plus d'informations là-dessus dans la prochaine section). Merci de noter qu'actuellement très peu de logiciels utilise la mémoire partagée POSIX. Donc, vous pouvez considérer le point de montage `/dev/shm` comme optionnel. Pour plus d'informations, voir `Documentation/filesystems/tmpfs.txt` dans le répertoire des sources du noyau.

7.16. Scripts de démarrage pour CLFS 2.0-pre2

Le paquet Bootscripts contient un ensemble de scripts de démarrage pour démarrer/arrêter le système CLFS lors de l'amorçage ou de l'arrêt.

7.16.1. Installation des scripts de démarrage

Installez le paquet :

```
make DESTDIR=${CLFS} install-minimal
```

Le script **setclock** lit le temps sur l'horloge matérielle, aussi connu sous le nom d'horloge BIOS ou CMOS (Complementary Metal Oxide Semiconductor). Si l'horloge matérielle est configurée en UTC, le 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.

Si vous ne savez pas si l'heure du système est configurée ou non sur UTC, vous pouvez le trouver avoir après démarré la nouvelle machine en lançant la commande **hwclock --localtime --show** et, si nécessaire, en éditant le script `/etc/sysconfig/clock`. Le pire qui pourrait se produire si vous vous trompez de diagnostic ici serait que l'heure affichée soit fausse.

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.

```
cat > ${CLFS}/etc/sysconfig/clock << "EOF"
# Début de /etc/sysconfig/clock

UTC=1

# Fin de /etc/sysconfig/clock
EOF
```

Les détails sur ce paquet sont situés dans Section 11.2.2, « Contenu des scripts de démarrage. »

7.17. Peupler /dev

7.17.1. Créez les nœuds de périphérique initiaux



Remarque

Vous devriez exécuter les commandes du reste de ce livre en tant qu'utilisateur `root`. Vérifiez que `${CLFS}` est configuré dans l'environnement de l'utilisateur `root` avant de continuer.

Quand le noyau démarre le système, il exige la présence de quelques nœuds de périphérique, en particulier les périphériques `console` et `null`. Nous allons créer les nœuds de périphérique sur le disque dur afin qu'ils soient disponibles avant qu'`udev` n'ait été démarré, et en plus quand Linux est démarré en mode monoutilisateur (d'où il résulte des droits restrictifs sur `console`). Créez ceux-ci en lançant les commandes suivantes :

```
mknod -m 600 ${CLFS}/dev/console c 5 1
mknod -m 666 ${CLFS}/dev/null c 1 3
```

Avant qu'`udev` ne démarre, un système de fichiers `tmpfs` est monté dans `/dev` et les entrées précédentes ne sont plus disponibles. La commande suivante crée des fichiers qui y sont copiés par le script de démarrage d'`udev` :

```
mknod -m 600 ${CLFS}/lib/udev/devices/console c 5 1
mknod -m 666 ${CLFS}/lib/udev/devices/null c 1 3
```

7.18. Changer de propriétaire

Actuellement, le répertoire `${CLFS}` et tous ses sous-répertoires appartiennent à l'utilisateur `clfs`, un utilisateur qui n'existe que sur le système hôte. Pour des raisons de sécurité, le répertoire racine `${CLFS}` et tous ses sous-répertoires devraient appartenir à `root`. Changez le propriétaire de `${CLFS}` et de ses sous-répertoires en lançant cette commande :

```
chown -Rv 0:0 ${CLFS}
```

Les fichiers suivants doivent appartenir au groupe `utmp` et non à `root`.

```
chgrp -v 13 ${CLFS}/var/run/utmp ${CLFS}/var/log/lastlog
```

7.19. Que faire ensuite

Maintenant, vous êtes arrivé au moment de copier votre répertoire `${CLFS}` vers votre machine cible. La méthode la plus simple serait de l'archiver et de copier le fichier.

```
tar -jcvf ${CLFS}.tar.bz2 ${CLFS}
```

Chapitre 8. Si vous allez vous chrooter

8.1. Introduction

Ce chapitre montre comment préparer une prison **chroot** pour y construire les paquets du système final.

8.2. Util-linux-2.22.1

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.

8.2.1. Installation d'Util-linux

Préparez la compilation d'Util-linux :

```
CC="${CC} ${BUILD64}" ./configure --prefix=/tools \  
  --build=${CLFS_HOST} --host=${CLFS_TARGET} \  
  --disable-makeinstall-chown --disable-login --disable-su \  
  --config-cache
```

Compilez le paquet :

```
make
```

Installez le paquet :

```
make install
```

Les détails sur ce paquet sont situés dans Section 10.30.3, « Contenu de Util-linux. »

8.3. Monter les systèmes de fichiers virtuels du noyau



Remarque

Vous devriez exécuter les commandes du reste de ce livre en tant qu'utilisateur `root`. Vérifiez que `${CLFS}` est configuré dans l'environnement de l'utilisateur `root` avant de continuer.

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

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

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

Maintenant, montez les systèmes de fichiers :

```
mount -vt proc proc ${CLFS}/proc
mount -vt sysfs sysfs ${CLFS}/sys
```

Rappelez-vous que si, pour une quelconque raison, vous vous arrêtez de travailler sur le système CLFS et recommencez plus tard, il est important de vérifier que ces systèmes de fichiers sont à nouveau montés avant d'entrer dans l'environnement chroot.

Deux nœuds de périphériques, `/dev/console` et `/dev/null`, doivent être présents sur le système de fichiers. Ils sont exigés par le noyau même avant le démarrage d'Udev très tôt dans le processus d'amorçage, donc nous les créons ici :

```
mknod -m 600 ${CLFS}/dev/console c 5 1
mknod -m 666 ${CLFS}/dev/null c 1 3
```

Une fois que le système est complet et qu'il démarre, le reste des nœuds de périphériques sont créés par le paquet Udev. Comme ce paquet n'est pas disponible pour nous maintenant, nous devons prendre en charge ces étapes pour fournir les nœuds de périphérique sur le système de fichiers CLFS. Nous allons utiliser l'option « `bind` » dans la commande `mount` pour faire apparaître la structure du `/dev` de notre système hôte dans le nouveau système de fichiers CLFS :

```
mount -v -o bind /dev ${CLFS}/dev
```

Des systèmes de fichiers supplémentaires seront bientôt montés à l'intérieur de l'environnement chroot. Pour maintenir l'hôte à jour, effectuez un « faux montage » pour chacun d'eux maintenant :

```
mount -f -vt tmpfs tmpfs ${CLFS}/dev/shm
mount -f -vt devpts -o gid=4,mode=620 devpts ${CLFS}/dev/pts
```

8.4. Entrer dans l'environnement Chroot

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

```
chroot "${CLFS}" /tools/bin/env -i \
HOME=/root TERM="${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 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 `CLFS` parce que tout le travail sera restreint au système de fichiers `CLFS`, car on a dit au shell Bash que `${CLFS}` 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**.

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 `proc` et `devpts` (traités dans la section précédente) et d'entrer de nouveau dans chroot avant de continuer les installations.

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

8.5. Changer de propriétaire



Remarque

Cette étape n'est pas facultative vu que certains binaires de `/tools` sont réglés sur `u+s`. Laisser les droits en l'état pourrait entraîner que certaines commandes, en particulier `mount`, échouent plus loin.

Pour l'instant, les répertoires `$CLFS/tools` et `/cross-tools` appartiennent à l'utilisateur `clfs`, un utilisateur qui n'existe que sur le système hôte. Bien que les répertoires `/tools` et `/cross-tools` puissent être effacés une fois que la construction du système `CLFS` est finie, vous pouvez les garder pour construire des systèmes `CLFS` supplémentaires. Si les répertoires `/tools` et `/cross-tools` restent 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 `/tools` et de tous les fichiers à l'intérieur, les exposant ainsi à des manipulations malintentionnées.

Pour éviter ce problème, vous pouvez ajouter l'utilisateur `clfs` au nouveau système `CLFS` plus tard lorsque vous créerez 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 des répertoires `/tools` et `/cross-tools` en les affectant à l'utilisateur `root` en exécutant les commandes suivantes :

```
chown -Rv 0:0 /tools
chown -Rv 0:0 /cross-tools
```

Les commandes utilisent `0:0` au lieu de `root:root` car **chown** n'est pas capable de résoudre le nom « root » jusqu'à ce que le fichier `passwd` a été créé.

8.6. Créer les répertoires

Il est temps de créer une structure sur le système de fichiers CLFS. Créez une arborescence de répertoires standard en lançant les commandes suivantes :

```
mkdir -pv /{bin,boot,dev,{etc/,}opt,home,lib{,64},mnt}
mkdir -pv /{proc,media/{floppy,cdrom},run/{,shm},sbin,src,sys}
mkdir -pv /var/{lock,log,mail,spool}
mkdir -pv /var/{opt,cache,lib{,64}/{misc,locate},local}
install -dv /root -m 0750
install -dv {/var,}/tmp -m 1777
mkdir -pv /usr/{,local/}{bin,include,lib{,64},sbin,src}
mkdir -pv /usr/{,local/}share/{doc,info,locale,man}
mkdir -pv /usr/{,local/}share/{misc,terminfo,zoneinfo}
mkdir -pv /usr/{,local/}share/man/man{1..8}
for dir in /usr{,/local}; do
    ln -sv share/{man,doc,info} $dir
done
install -dv /usr/lib/locale
ln -sv ../lib/locale /usr/lib64
```

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 ont été 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.

8.6.1. 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 l'arborescence créée ci-dessus, 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.

8.7. Créer les liens symboliques essentiels

Certains programmes utilisent des chemins liés en dur à des programmes qui n'existent pas encore. Afin de satisfaire ces programmes, créez un certain nombre de liens symboliques qui seront remplacés par des fichiers réels tout au long du chapitre suivant après que le logiciel a été installé.

```
ln -sv /tools/bin/{bash,cat,echo,grep,pwd,stty} /bin
ln -sv /tools/bin/file /usr/bin
ln -sv /tools/lib/libgcc_s.so{,.1} /usr/lib
ln -sv /tools/lib64/libgcc_s.so{,.1} /usr/lib64
ln -sv /tools/lib/libstd* /usr/lib
ln -sv /tools/lib64/libstd* /usr/lib64
ln -sv bash /bin/sh
ln -sv /run /var/run
```

8.8. Options de construction

Nous aurons besoin de paramétrer les options spécifiques à la cible pour le compilateur et les éditeurs de liens :

```
export BUILD32="-m32"
export BUILD64="-m64"
```

Vous aurez besoin de paramétrer la triplette cible de votre hôte pour 32 bits :

```
export CLFS_TARGET32="i686-pc-linux-gnu"
```

Pour éviter des erreurs lorsque vous reviendrez à votre construction, nous allons exporter ces variables pour empêcher toute erreur de construction dans le futur :

```
cat >> ${CLFS}/root/.bash_profile << EOF
export BUILD32="${BUILD32}"
export BUILD64="${BUILD64}"
export CLFS_TARGET32="${CLFS_TARGET32}"
EOF
```

8.9. Créer le mot de passe, le groupe et les fichiers journal

Afin que l'utilisateur `root` puisse se connecter et pour que le nom « `root` » soit reconnu, il doit y avoir des entrées adéquates 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
EOF
```

Le mot de passe actuel pour `root` (le « `x` » utilisé ici n'est qu'un paramètre fictif) sera réglé plus tard.

Utilisateurs supplémentaires que vous pourriez vouloir ajouter :

```
bin:x:1:1:bin:/bin:/bin/false
```

Peut être utile pour la compatibilité avec des applications héritées.

```
daemon:x:2:6:daemon:/sbin:/bin/false
```

Il est souvent recommandé d'utiliser l'ID d'un groupe ou d'un utilisateur non privilégiés pour l'exécution de démons, afin de limiter leur accès au système.

```
adm:x:3:16:adm:/var/adm:/bin/false
```

Était utilisé pour des programmes qui effectuaient des tâches d'administration.

```
lp:x:10:9:lp:/var/spool/lp:/bin/false
```

Utilisé par des programmes pour l'impression

```
mail:x:30:30:mail:/var/mail:/bin/false
```

Souvent utilisé par des programmes de messagerie

```
news:x:31:31:news:/var/spool/news:/bin/false
```

Souvent utilisé pour un réseau de serveurs de nouvelles (*news*)

```
operator:x:50:0:operator:/root:/bin/bash
```

Souvent utilisé pour permettre aux opérateurs du système d'accéder au système

```
postmaster:x:51:30:postmaster:/var/spool/mail:/bin/false
```

Utilisé généralement comme compte qui reçoit toutes les informations de problèmes avec le serveur de messagerie


```
nobody:x:65534:65534:nobody:/:/bin/false
```

Utilisé par NFS

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

```
cat > /etc/group << "EOF"
root:x:0:
bin:x:1:
sys:x:2:
kmem:x:3:
tty:x:4:
tape: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:
EOF
```

Groupes supplémentaires que vous pourriez vouloir ajouter

```
adm:x:16:root,adm,daemon
```

Tous les utilisateurs de ce groupe ont le droit de faire des tâches d'administration

```
console:x:17:
```

Ce groupe a un accès direct à la console

```
cdrw:x:18:
```

Ce groupe est autorisé à utiliser le lecteur CDRW

```
mail:x:30:mail
```

Utilisé par MTAs (Mail Transport Agents)

```
news:x:31:news
```

Utilisé par le réseau de serveurs de nouvelles

```
users:x:1000:
```

Le GID utilisé par défaut par shadow pour les nouveaux utilisateurs

```
nogroup:x:65533:
```

C'est le groupe par défaut utilisé par certains programmes qui n'ont pas besoin d'un groupe

```
nobody:x:65534:
```

C'est utilisé par NFS

Les groupes créés ne font partie d'aucun standard—ce sont des groupes décidés d'une part par les exigences de la configuration d'Udev dans le système final, d'autre part par la convention couramment utilisée par un grand nombre de distributions Linux existantes. La *Linux Standard Base* (LSB, disponible sur <http://www.linuxbase.org>) recommande uniquement que, après le groupe « root » ayant l'identifiant de groupe (GID) 0, un groupe « bin » avec un GID de 1 soit présent. L'administrateur système peut choisir librement tout autre noms de groupe et GIDs, vu que les programmes bien écrits ne dépendent pas des numéros GID mais utilisent plutôt le nom d'un groupe.

Pour supprimer l'invite « I have no name! » démarrez un nouveau shell. Puisqu'on a installé une Glibc complète dans le Constructing Cross-Compile Tools et que les répertoires `/etc/passwd` et `/etc/group` ont été créés, la résolution des noms d'utilisateur et de groupe va à présent fonctionner.

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

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

Les programmes **login**, **agetty** et **init** (et d'autres) utilisent un certain nombre de fichiers journal pour enregistrer des informations telles que ceux qui se sont connectés au système et quand. Néanmoins, ces programmes n'écriront pas dans les fichiers journal s'ils n'existent pas déjà. Initialisez les fichiers journal et donnez-leur les bons droits :

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

Le fichier `/var/run/utmp` enregistre les utilisateurs actuellement connectés. Le fichier `/var/log/wtmp` enregistre toutes les connexions et les déconnexions. Le fichier `/var/log/lastlog` enregistre le moment où chaque utilisateur s'est connecté pour la dernière fois. Le fichier `/var/log/btmp` enregistre les tentatives de connexion erronées.

8.10. Monter les systèmes de fichiers du noyau

8.10.1. Mounter les systèmes de fichiers du noyau supplémentaires

Montez les bons systèmes de fichiers virtuels (du noyau) dans les répertoires récemment créés :

```
mount -vt devpts -o gid=4,mode=620 none /dev/pts
mount -vt tmpfs none /dev/shm
```

Il se peut que les commandes **mount** exécutées ci-dessus produisent le message d'avertissement suivant :

```
can't open /etc/fstab: No such file or directory.
```

Ce fichier—`/etc/fstab`—n'a pas encore été créé mais il n'est pas non plus nécessaire pour que les systèmes de fichiers soient montés correctement. Tel quel, vous pouvez ignorer l'avertissement en toute sécurité.

Partie V. Construction du système CLFS

Chapitre 9. Construction des outils de test

9.1. Introduction

Ce chapitre construit les outils nécessaires à certains paquets pour exécuter les tests que comportent les paquets, par exemple **make check**. Tcl, Expect et DejaGNU sont nécessaires pour les suites de tests de GCC et de Binutils. Il se peut que l'installation de trois paquets pour des tests semble excessive, mais c'est très rassurant, sinon essentiel, de savoir que les outils les plus importants fonctionnent correctement.

9.2. Tcl-8.5.12

Le paquet Tcl contient le *Tool Command Language*.

9.2.1. Installation de Tcl

Préparez la compilation de Tcl :

```
cd unix
CC="gcc ${BUILD64}" ./configure --prefix=/tools --libdir=/tools/lib64
```

Compilez le paquet :

```
make
```

Installez le paquet :

```
make install
```

Les fichiers d'en-tête privés de Tcl sont nécessaires pour le paquet suivant, Expect. Installez-les dans /tools :

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

Créez maintenant un lien symbolique nécessaire :

```
ln -sv tclsh8.4 /tools/bin/tclsh
```

9.2.2. Contenu de Tcl

Programmes installés: tclsh (lien vers tclsh8.4) et tclsh8.4

Bibliothèques installées: libtcl8.5.so, libtclstub8.5.a

Descriptions courtes

tclsh8.5	Le shell de commandes Tcl
tclsh	Un lien vers tclsh8.4
libtcl8.5.so	La bibliothèque Tcl
libtclstub8.5.a	La bibliothèque Stub de Tcl

9.3. Expect-5.45

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

9.3.1. Installation de Expect

Maintenant, préparez la compilation d'Expect :

```
CC="gcc ${BUILD64}" ./configure --prefix=/tools \
  --with-tcl=/tools/lib64 --with-tclinclude=/tools/include \
  --libdir=/tools/lib64
```

Voici la signification des options de configure :

`--with-tcl=/tools/lib64`

Cela assure que le script configure cherche l'installation de Tcl à l'endroit des outils temporaires.

`--with-tclinclude=/tools/include`

Ceci dit explicitement à Expect où trouver les en-têtes internes de Tcl. L'utilisation de cette option évite les conditions où **configure** échoue car il ne peut pas découvrir automatiquement l'emplacement du répertoire source de Tcl.

Compilez le paquet :

```
make
```

Installez le paquet :

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

Voici la signification du paramètre de make :

`SCRIPTS=""`

Ceci empêche l'installation des scripts expect supplémentaires dont on n'a pas besoin.

9.3.2. Contenu d'Expect

Programme installé:	expect
Répertoire installé:	libexpect-5.43.a

Courte description

expect	Communique avec les autres programmes interactifs suivant un script.
libexpect-5.43.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)

9.4. DejaGNU-1.5

Le paquet DejaGNU contient une chaîne d'outils pour tester d'autres programmes.

9.4.1. Installation de DejaGNU

Préparez la compilation de DejaGNU :

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

Compilez et installez le paquet :

```
make install
```

9.4.2. Contenu de DejaGNU

Programme installé: runtest

Descriptions courtes

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

Chapitre 10. Installation des logiciels du système de base

10.1. Introduction

Dans ce chapitre, nous entrons dans l'espace de construction et nous commençons à construire sérieusement le système CLFS. L'installation de ce logiciel est simple. Bien que 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 l'utilisateur (ou le système) en a besoin. Pour chaque paquet installé, un résumé de son contenu est donné, suivi par des descriptions concises de chaque programme et de chaque bibliothèque que le paquet a installé.

En utilisant les optimisations du compilateur, merci de lire l'astuce sur l'optimisation sur <http://lfs.traduc.org/view/astuces/optimization-fr.txt>. Les optimisations du compilateur peuvent faire qu'un programme s'exécute un peu plus rapidement mais elles peuvent aussi causer des difficultés et des problèmes de compilation 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 CLFS 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 les paquets en parallèle. La compilation en parallèle permet de gagner du temps (tout particulièrement sur les machines à plusieurs CPU), mais cela pourrait résulter en un programme contenant un chemin codé en dur vers `/tools`, ce qui empêchera le programme de fonctionner si ce répertoire est supprimé.

Pour garder une trace des fichiers installés par un paquet particulier, vous pouvez utiliser un gestionnaire de paquets. Pour une vue générale des différents types de gestionnaires de paquets, jetez un œil sur la page suivante.

10.2. Gestion de paquets

La gestion de paquets est un ajout souvent demandé au livre CLFS. 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 gestionnaire de paquets parfait 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 CLFS ou CBLFS :

- 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 des 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.

10.2.1. Problèmes de mise à jour

Un gestionnaire de paquets facilite la mise à jour des nouvelles versions au moment de leur publication. Généralement, les instructions des livres CLFS et CBLFS 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 vers une nouvelle version mineure, de reconstruire CLFS. 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 mis à jour et si le nom de cette dernière est modifié, alors les paquets liés dynamiquement à la bibliothèque devront être recompilés pour être liés à la nouvelle bibliothèque. (Remarquez qu'il n'y a aucune corrélation entre la version du paquet et le nom de la bibliothèque.) Par exemple, considérez un paquet foo-1.2.3 qui installe une bibliothèque partagée de nom `libfoo.so.1`. Disons que vous mettez à jour le paquet avec une nouvelle version foo-1.2.4 qui installe une bibliothèque partagée de nom `libfoo.so.2`. Dans ce cas, tous les paquets liés dynamiquement à `libfoo.so.1` doivent être recompilés pour être liés à `libfoo.so.2`. Remarquez que vous ne devez pas supprimer les anciennes bibliothèques jusqu'à ce que les paquets indépendants soient recompilés.
- Si vous mettez à jour un système en cours d'exécution, soyez très attentif avec les paquets qui utilisent **cp** au lieu de **install** pour installer les fichiers. La deuxième commande est généralement plus sûre si l'exécutable ou la bibliothèque est déjà chargé en mémoire.

10.2.2. Techniques de gestion de paquets

Ce qui suit est une liste des 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.

10.2.2.1. Tout est dans ma tête !

Oui, c'est une technique de gestion de paquets. Certains n'éprouvent pas le besoin d'un gestionnaire de paquets parce qu'ils connaissent très bien les paquets et connaissent les fichiers installés par chaque paquet. Certains utilisateurs n'en ont pas besoin parce qu'ils planifient la reconstruction entière de LFS lorsqu'un paquet est modifié.

10.2.2.2. Installer dans des répertoires séparés

C'est une gestion des paquets tellement simple qu'elle ne nécessite aucun paquet supplémentaire pour gérer les installations. Chaque paquet est installé dans un répertoire séparé. Par exemple, le paquet foo-1.1 est installé dans `/usr/pkg/foo-1.1` et un lien symbolique est créé 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.

10.2.2.3. Gestion de paquet par lien symbolique

C'est une variante de la technique précédente. Chaque paquet est installé de façon similaire au schéma précédent. Mais au lieu de créer 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 puissent 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 se lier à `libfoo` comme vous vous y attenderiez. 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`.

10.2.2.4. Basé sur un horodatage

Avec cette technique, un fichier est horodaté 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 horodaté n'a été créé. `install-log` est un gestionnaire de paquets écrit avec cette approche.

Bien que ce schéma ait l'avantage d'être simple, il a deux inconvénients. Si à l'installation, les fichiers sont installés sans être horodatés avec 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.

10.2.2.5. Basée sur LD_PRELOAD

Dans cette approche, une bibliothèque est préchargée avant l'installation. Pendant l'installation, cette bibliothèque poursuit les paquets qui vont être installés en s'attachant à divers exécutables tels que **cp**, **install**, **mv** et en surveillant les appels du système qui modifient le système de fichiers. Pour que cette approche fonctionne, tous les exécutables doivent être liés de façon dynamique sans le bit `suid` ou `sgid`. Il se peut que le préchargement de la bibliothèque provoque des effets secondaires non souhaités pendant l'installation. Donc, il est conseillé d'effectuer des tests pour s'assurer que le gestionnaire de paquets ne casse rien et enregistre tous les fichiers appropriés.

10.2.2.6. Créer des archives de paquets

Dans ce schéma, l'installation d'un paquet est faussée dans un répertoire séparé comme décrit plus haut. Après l'installation, une archive du paquet est créée en utilisant les fichiers installés. L'archive est ensuite utilisée pour installer le paquet soit sur la machine locale soit même 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 Portage de Gentoo. Une astuce décrivant comment adopter ce style de gestion de paquets pour les systèmes CLFS se trouve à <http://hints.cross-lfs.org/index.php/Fakeroot>.

10.3. À nouveau à propos des suites de tests

Dans la construction du système final, vous n'effectuez plus une compilation croisée donc il est possible de lancer les suites de test des paquets. Certaines des suites de tests sont plus importantes que d'autres. Par exemple, les suites de tests des paquets formant le cœur de l'ensemble des outils—GCC, Binutils et Glibc—sont de la plus grande importance étant donné leur rôle central dans un système fonctionnel. Les suites de tests pour GCC et Glibc peuvent prendre beaucoup de temps pour se terminer, surtout sur du matériel lent, mais elles sont fortement recommandées.

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 (si vous êtes chrooté) le système hôte ne dispose pas du système de fichiers `devpts` configuré correctement. Ce problème est traité avec plus de détails dans sur <http://trac.cross-lfs.org/wiki/faq#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://cross-lfs.org/testsuite-logs/git/> pour vérifier si ces échecs sont attendus. Ce site est valide pour tous les tests effectués dans ce livre.

10.4. Perl-5.16.2 temporaire

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

10.4.1. Installation de Perl

Tout d'abord, modifiez certains chemins vers la bibliothèque C codés en dur en appliquant le correctif suivant :

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

Modifiez un chemin lié en dur de `/usr/include` en `/tools/include` :

```
sed -i 's@/usr/include@/tools/include at g' ext/Errno/Errno_pm.PL
```

Préparez la compilation de Perl temporaire ::

```
./configure.gnu --prefix=/tools -Dcc="gcc ${BUILD32}"
```

Voici la signification des options de configure :

```
-Dcc="gcc"
```

Dit à Perl d'utiliser **gcc** à la place du **cc** par défaut.

Compilez le paquet :

```
make
```

Bien que Perl soit fourni avec une suite de tests, il n'est pas recommandé de l'exécuter à ce moment, vu que cette installation de Perl n'est que temporaire. Vous pouvez lancer la suite de tests plus tard dans ce chapitre si vous le souhaitez.

Installez le paquet :

```
make install
```

Enfin, créez un lien symbolique nécessaire :

```
ln -sfv /tools/bin/perl /usr/bin
```

Les détails sur ce paquet sont disponibles dans Section 10.47.2, « Contenu de Perl. »

10.5. Linux-Headers-3.4.17

Le noyau Linux contient une cible make qui installe des en-têtes du noyau « propres ».

10.5.1. Installation de Linux-Headers

Pour cette étape, vous aurez besoin de l'archive tar du noyau.

Installez les fichiers d'en-tête du noyau :

```
make mrproper
make headers_check
make INSTALL_HDR_PATH=dest headers_install
cp -rv dest/include/* /usr/include
find /usr/include -name .install -or -name ..install.cmd | xargs rm -fv
```

Voici la signification des commandes :

make mrproper

S'assure que le répertoire des sources du noyau est propre.

make ARCH=x86_64 headers_check

Nettoie les en-têtes raw du noyau afin qu'elles puissent être utilisées par les programmes d'espace utilisateur.

make ARCH=x86_64 INSTALL_HDR_PATH=dest headers_install

La cible headers_install supprime normalement tout le répertoire de destination (par défaut /usr/include) avant d'installer les en-têtes. Pour empêcher cela, nous disons au noyau d'installer les en-têtes dans un autre répertoire à l'intérieur de celui des sources.

find /usr/include -name .install -or -name ..install.cmd | xargs rm -fv

Supprime un certain nombre de fichiers de débogage inutiles qui ont été installés.

10.5.2. Contenu de Linux-Headers

En-têtes installées: /usr/include/{asm,asm-generic,drm,linux,mtd,rdma,scsi,sound,video,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,asm-generic,drm,linux,mtd,rdma,sound,video}/*.
*.h Les en-têtes Linux API

10.6. Man-pages-3.43

Le paquet Man-pages contient environ 1 200 pages de manuel.

10.6.1. Installation de Man-pages

Installez Man-pages en lançant :

```
make install
```

10.6.2. Contenu de Man-pages

Fichiers installés: Diverses pages de man

Descriptions courtes

pages man Ce paquet contient des pages de man qui décrivent ce qui suit : Les en-têtes POSIX (section 0p), Les outils POSIX (section 1p), POSIX functions (section 3p), Les commandes utilisateur (section 1), system calls (section 2), Les appels libc (section 3), device information (section 4), Les formats de fichier (section 5), games (section 6), Les conventions et des macro paquet (section 7), L'administration système (section 8) et Le noyau (section 9).

10.7. EGLIBC-2.15 32 bit Libraries

Le paquet EGLIBC contient la bibliothèque C principale. Cette bibliothèque fournit toutes les routines de base pour allouer de la mémoire, rechercher dans des répertoires, ouvrir et fermer des fichiers, les lire et les écrire, gérer les chaînes, faire de la recherche de motifs, faire de l'arithmétique etc.

10.7.1. Installation de EGLIBC



Remarque

Certains paquets non compris dans CLFS suggèrent d'installer GNU libiconv pour traduire les données d'un encodage en un autre. La page d'accueil du projet (<http://www.gnu.org/software/libiconv/>) précise que « 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. » EGLIBC fournit une implémentation d'`iconv()` et peut convertir de l'Unicode, du coup libiconv n'est pas requis sur un système CLFS.

En multilib, on a tendance à penser que compiler pour `${CLFS_TARGET32}` n'est *pas* de la compilation croisée. EGLIBC a une vision traditionnelle selon laquelle si vous construisez pour un autre hôte, vous faites une compilation croisée, donc vous n'exécuterez pas les tests, et donc vous n'avez pas besoin des fichiers de locale. Quand on lance les tests, beaucoup échoueront sans les fichiers de localisation. Le sed suivant permet aux tests de réussir :

```
sed -i '/cross-compiling/s@ifeq@ifneq@g' ../eglibc-2.15/localedata/Makefile
```

Ce même script effectue ses tests en essayant de compiler des programmes de test contre certaines bibliothèques. Cependant, il ne spécifie pas le `ld.so`, or notre ensemble d'outils est configuré pour utiliser celui de `/tools`. L'ensemble de commandes suivant obligera le script à utiliser le chemin complet du nouveau `ld.so` qu'on vient d'installer :

```
LINKER=$(readelf -l /tools/bin/bash | sed -n 's@.*interpret.*/tools\(.*\)]$@\1'p)
sed -i "s|libs -o|libs -L/usr/lib -Wl,-dynamic-linker=${LINKER} -o|" \
    scripts/test-installation.pl
unset LINKER
```

Le correctif suivant corrige un problème qui peut faire planter ALSA :

```
patch -Np1 -i ../eglibc-2.15-fixes-1.patch
```

Le système de construction d'EGLIBC est autosuffisant et s'installe parfaitement, même si notre fichier specs pour le compilateur et l'éditeur de liens pointent toujours vers `/tools`. Les specs et l'éditeur de liens ne peuvent pas être ajustés avant l'installation de la EGLIBC parce que les tests d'autoconf d'EGLIBC donneraient alors des résultats faussés, défaussant ainsi notre but d'achever une construction propre.

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

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

Préparez la compilation d'EGLIBC :

```
CC="gcc ${BUILD32}" CXX="g++ ${BUILD32}" \
CFLAGS="-march=$(cut -d- -f1 <<< ${CLFS_TARGET32}) \
-mtune=generic -g -O2" ../eglibc-2.15/configure --prefix=/usr \
--disable-profile --enable-kernel=2.6.32 \
--libexecdir=/usr/lib/eglibc --host=${CLFS_TARGET32}
```

Voici la signification de la nouvelle option de configure :

```
--libexecdir=/usr/lib/glibc
```

Ceci change l'emplacement du programme **pt_chown** de celui par défaut `/usr/libexec` vers `/usr/lib/glibc`.

Compilez le paquet :

```
make
```



Important

La suite de tests d'EGLIBC est considérée comme critique. Ne la sautez sous aucun prétexte.

À la fin de l'installation, le système de construction exécutera un test de propreté pour s'assurer que tout s'est installé correctement. Ce script essaiera de tester la présence d'une bibliothèque utilisée uniquement pour la suite de tests, et elle n'est jamais installée. Empêchez le script de tester la présence de cette bibliothèque avec la commande suivante :

```
sed -i 's/\(&& $name ne\) "db1"/ & \1 "nss_test1"/' scripts/test-installation.
```

Testez les résultats :

```
cp -v ../eglibc-2.15/iconvdata/gconv-modules iconvdata
make -k check 2>&1 | tee eglibc-check-log; grep Error eglibc-check-log
```

La suite de tests EGLIBC est très dépendante de certaines fonctions du système hôte, en particulier du noyau. Normalement, le test `posix/annexc` échoue et vous devriez voir `Error 1 (ignored)` dans la sortie. Excepté cela, la suite de tests d'EGLIBC devrait toujours passer. Néanmoins, dans certaines circonstances, certains échecs sont inévitables. Si un test échoue à cause d'un programme manquant (ou d'un lien symbolique manquant), ou du fait d'une erreur de segmentation, vous verrez un code d'erreur supérieur à 127 et les détails seront dans le journal. De manière plus générale, les tests échoueront avec `Error 2` - pour eux le contenu du fichier `.out`, comme `posix/annexc.out` peut vous donner des informations. Voici une liste des problèmes les plus fréquents :

- Les tests *math* échouent quelque fois. Certaines optimisations sont connues pour être une cause de cela.
- Si vous avez monté la partition CLFS avec l'option *noatime*, le test *atime* échouera. Comme mentionné dans Section 2.4, « Monter la nouvelle partition », n'utilisez pas l'option *noatime* lors de la construction de CLFS.
- Lors d'une exécution sur un matériel ancien et lent, quelques tests peuvent échouer à cause de délais dépassés.

Bien que ce ne soit qu'un simple message, l'étape d'installation de EGLIBC 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
```


Les détails sur ce paquet sont disponibles dans Section 10.8.5, « Contenu d'EGLIBC. »

10.8. EGLIBC-2.15 64 bits

Le paquet EGLIBC contient la bibliothèque C principale. Cette bibliothèque fournit toutes les routines de base pour allouer de la mémoire, rechercher dans des répertoires, ouvrir et fermer des fichiers, les lire et les écrire, gérer les chaînes, faire de la recherche de motifs, faire de l'arithmétique etc.

10.8.1. Installation d'EGLIBC

À la fin de l'installation, le système de construction exécutera un test de propreté pour s'assurer que tout s'est installé correctement. Ce script essaiera de tester la présence d'une bibliothèque utilisée uniquement pour la suite de tests, et elle n'est jamais installée. Empêchez le script de tester la présence de cette bibliothèque avec la commande suivante :

```
sed -i 's/\(&& $name ne\) "dbl"/ & \1 "nss_test1"/' scripts/test-installation.
```

Ce même script effectue ses tests en essayant de compiler des programmes de test contre certaines bibliothèques. Cependant, il ne spécifie pas le ld.so, or notre ensemble d'outils est configuré pour utiliser celui de /tools. L'ensemble de commandes suivant obligera le script à utiliser le chemin complet du nouveau ld.so qu'on vient d'installer :

```
LINKER=$(readelf -l /tools/bin/bash | sed -n 's@.*interpret.*/tools\(.*\)]$@\1' |  
sed -i 's|libs -o|libs -L/usr/lib64 -Wl,-dynamic-linker=${LINKER} -o|" \  
scripts/test-installation.pl  
unset LINKER
```

Le correctif suivant corrige un problème qui peut faire planter ALSA :

```
patch -Np1 -i ../eglibc-2.15-fixes-1.patch
```

Le système de construction d'EGLIBC est autosuffisant et s'installe parfaitement, même si notre fichier specs pour le compilateur et l'éditeur de liens pointent toujours vers /tools. Les specs et l'éditeur de liens ne peuvent pas être ajustés avant l'installation de la EGLIBC parce que les tests d'autoconf d'EGLIBC donneraient alors des résultats faussés, défaussant ainsi notre but d'achever une construction propre.

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

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

Demande à EGLIBC d'installer ces bibliothèques 64 bits dans /lib64:

```
echo "slibdir=/lib64" >> configparms
```

Préparez la compilation d'EGLIBC :

```
CC="gcc ${BUILD64}" CXX="g++ ${BUILD64}" \  
CFLAGS="-mtune=generic -g -O2" \  
../eglibc-2.15/configure --prefix=/usr \  
--disable-profile --enable-kernel=2.6.32 \  
--libexecdir=/usr/lib64/eglibc --libdir=/usr/lib64
```

La signification des nouvelles options de configuration:

```
--libexecdir=/usr/lib64/glibc
```

This changes the location of the **pt_chown** program from its default of /usr/libexec to /usr/lib64/glibc.

Compilez le paquet :

```
make
```



Important

La suite de tests d'EGLIBC est considérée comme critique. Ne la sautez sous aucun prétexte.

Testez les résultats :

```
cp -v ../eglibc-2.15/iconvdata/gconv-modules iconvdata
make -k check 2>&1 | tee eglibc-check-log; grep Error eglibc-check-log
```

La suite de tests EGLIBC est très dépendante de certaines fonctions du système hôte, en particulier du noyau. Normalement, le test `posix/annexc` échoue et vous devriez voir `Error 1 (ignored)` dans la sortie. Excepté cela, la suite de tests d'EGLIBC devrait toujours passer. Néanmoins, dans certaines circonstances, certains échecs sont inévitables. Si un test échoue à cause d'un programme manquant (ou d'un lien symbolique manquant), ou du fait d'une erreur de segmentation, vous verrez un code d'erreur supérieur à 127 et les détails seront dans le journal. De manière plus générale, les tests échoueront avec `Error 2` - pour eux le contenu du fichier `.out`, comme `posix/annexc.out` peut vous donner des informations. Voici une liste des problèmes les plus fréquents :

- Les tests *math* échouent quelque fois. Certaines optimisations sont connues pour être une cause de cela.
- Si vous avez monté la partition CLFS avec l'option *noatime*, le test *atime* échouera. Comme mentionné dans Section 2.4, « Monter la nouvelle partition », n'utilisez pas l'option *noatime* lors de la construction de CLFS.
- Lors d'une exécution sur un matériel ancien et lent, quelques tests peuvent échouer à cause de délais dépassés.

Installez le paquet :

```
make install
```

Installez les en-têtes liées à NIS et RPC qui ne sont pas installées par défaut.

```
cp -v ../eglibc-2.15/sunrpc/rpc/*.h /usr/include/rpc
cp -v ../eglibc-2.15/sunrpc/rpcsvc/*.h /usr/include/rpcsvc
cp -v ../eglibc-2.15/nis/rpcsvc/*.h /usr/include/rpcsvc
```

10.8.2. Internationalisation

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.

```
make localedata/install-locales
```

Pour gagner du temps, une alternative au lancement de la commande précédente (qui génère et installe toutes les locales listées dans le fichier `EGLIBC-2.15/localedata/SUPPORTED` est de n'installer que les locales nécessaires et exigées. Vous pouvez faire cela avec la commande **localedef**. Des informations sur cette commande se trouvent dans le fichier `INSTALL` des sources d'EGLIBC. Néanmoins, il y a un nombre de locales essentielles

pour que les tests des paquets à venir passent, en particulier les tests *libstdc++* de GCC. Les instructions suivantes au lieu de la cible *install-locale* utilisée ci-dessus, installeront l'ensemble minimal des locales nécessaires pour que les tests s'exécutent avec succès :

```
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 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 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 it_IT -f ISO-8859-1 it_IT
localedef -i ja_JP -f EUC-JP ja_JP
```

Certaines locales installées par la commande **make localedata/install-locales** ci-dessus ne sont pas supportées correctement par certaines applications qui sont dans CLFS et CBLFS. À la vue des divers problèmes survenus du fait de certains présupposés des programmeurs de certaines applications, lesquelles se cassent dans de telles locales, vous ne devriez pas utiliser CLFS dans des locales utilisant des encodages multioctets (y compris UTF-8) ou le sens d'écriture de droite à gauche. De nombreux correctifs non officiels et instables sont nécessaires pour corriger ces problèmes et les développeurs de CLFS ont décidé de ne pas supporter des locales complexes pour l'instant. Ceci s'applique aux locales *ja_JP* et *fa_IR* — elles n'ont été installées que pour que les tests de GCC et de Gettext passent et le programme **watch** (qui fait partie du paquet Procps) ne fonctionne pas correctement avec elles. Diverses solutions pour contourner ces restrictions sont documentées dans les astuces liées à l'internationalisation.

10.8.3. Configurer EGLIBC

Le fichier */etc/nsswitch.conf* doit être créé car, bien que EGLIBC en fournisse un par défaut lorsque ce fichier est manquant ou corrompu, les valeurs par défaut d'EGLIBC 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 exécutant ce qui suit :

```
cat > /etc/nsswitch.conf << "EOF"
# Début de /etc/nsswitch.conf

passwd: files
group: files
shadow: files

hosts: files dns
networks: files

protocols: files
services: files
ethers: files
rpc: files

# Fin de /etc/nsswitch.conf
EOF
```

Pour déterminer dans quel fuseau horaire vous vous situez, exécutez le script suivant :

tzselect

Après avoir répondu à quelques questions sur votre localisation, le script affichera le nom du fuseau horaire (quelque chose comme *EST5EDT* ou *Canada/Eastern*). 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 *Canada/Eastern*).

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 d'anticiper la situation où `/usr` est une partition séparée. Ceci pourrait être important en démarrant en mode utilisateur unique.

10.8.4. Configurer le chargeur dynamique

Par défaut, le chargeur dynamique (`/lib/ld-linux.so.2` pour les exécutables 32 bits (et `/lib64/ld-linux-x86-64.so.2` pour les exécutables 64 bits) cherche dans `/lib`, `/lib64`, `/usr/lib` et `/usr/lib64` les bibliothèques dynamiques exigées par les programmes lorsqu'ils sont exécutés. Cependant, s'il y a des bibliothèques dans d'autres répertoires que ceux-là, ils doivent être ajoutés au fichier `/etc/ld.so.conf` afin que le chargeur dynamique les trouve. Les répertoires suivants sont connus pour contenir des bibliothèques : `/usr/local/lib`, `/usr/local/lib64`, `/opt/lib` et `/opt/lib64`. Ajoutez donc ces répertoires au chemin de recherche du chargeur dynamique.

Create a new file `/etc/ld.so.conf` by running the following:

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

/usr/local/lib
/usr/local/lib64
/opt/lib
/opt/lib64

# End /etc/ld.so.conf
EOF
```

10.8.5. Contenu d'EGLIBC

Programmes installés:	catchsegv, gencat, getconf, getent, iconv, iconvconfig, ldconfig, ldd, lddlibc4, locale, localedef, makedb, mtrace, nsd, pcprofiledump, pldd, pt_chown, rpcgen, sln, sprof, tzselect, xtrace, zdump, and zic
Bibliothèques installées:	ld.so, libBrokenLocale.[a,so], libSegFault.so, libanl.[a,so], libbsd-compat.a, libc.[a,so], libc_nonshared.a, libcidn.[a,so], libcrypt.[a,so], libdl.[a,so], libg.a, libieee.a, libm.[a,so], libmcheck.a, libmemusage.so, libnsl.a, 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/gconv, /usr/lib/eglibc, /usr/lib/locale, /usr/share/i18n, /usr/share/zoneinfo, /var/cache/ldconfig

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 la configuration système pour les variables spécifiques à un système de fichiers
getent	Récupère les entrées à partir d'une base de données d'administration
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	Dit au compilateur d'activer ou de désactiver l'utilisation des locales POSIX pour les opérations construites en dur
localedef	Compile les spécifications de locale
makedb	Crée une base de données 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
nsd	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 partagés dynamiques utilisés en lançant 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 (<i>Remote Procedure Call</i>)
sln	Un programme lié statiquement qui crée des liens symboliques

sotruss	Trace les appels de procédures de bibliothèque partagée d'une commande spécifiée
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
<code>ld.so</code>	Le programme d'aide des bibliothèques partagées exécutables
<code>libBrokenLocale</code>	Utilisée par des programmes comme Mozilla pour résoudre des locales cassées
<code>libSegFault</code>	Le gestionnaire de signaux d'erreurs de segmentation
<code>libanl</code>	Une bibliothèque asynchrone de recherche de noms
<code>libbsd-compat</code>	Fournit la portabilité nécessaire pour faire fonctionner certains programmes BSD (Berkeley Software Distribution) sous Linux
<code>libc</code>	La principale bibliothèque C
<code>libcidn</code>	Utilisé en interne par EGLIBC pour la gestion des noms de domaine internationaux dans la fonction <code>getaddrinfo()</code>
<code>libcrypt</code>	La bibliothèque de chiffrement
<code>libdl</code>	La bibliothèque de l'interface du chargeur dynamique
<code>libg</code>	Une bibliothèque d'exécution en cours pour g++
<code>libieee</code>	La bibliothèque de points flottants de <i>Institute of Electrical and Electronic Engineers</i> (IEEE).
<code>libm</code>	La bibliothèque mathématique
<code>libmcheck</code>	Contient du code exécuté au démarrage
<code>libmemusage</code>	Utilisé par memusage (compris dans EeGLIBC mais pas compilé dans un système CLFS de base car il a des dépendances supplémentaires) pour aider à la récupération d'informations sur l'utilisation de la mémoire par un programme
<code>libnsl</code>	La bibliothèque de services réseau
<code>libnss</code>	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
<code>libpcprofile</code>	Contient des fonctions de profilage utilisées pour tracer le temps CPU dépensé sur les lignes de code source
<code>libpthread</code>	La bibliothèque threads POSIX
<code>libresolv</code>	Contient des fonctions de création, d'envoi et d'interprétation de paquets pour les serveurs de noms de domaine Internet
<code>librpcsvc</code>	Contient des fonctions apportant différents services RPC
<code>librt</code>	Contient des fonctions fournissant la plupart des interfaces spécifiées par l'extension temps réel de POSIX.1b
<code>libthread_db</code>	Contient des fonctions utiles pour construire des débogueurs de programmes multi-threads
<code>libutil</code>	Contient du code pour les fonctions « standards » utilisées par de nombreux outils Unix

10.9. Ajuster la chaîne d'outils

Maintenant, on modifie le fichier de specs de GCC pour qu'il pointe vers le nouvel éditeur de liens dynamique. Une commande **perl** s'en charge :

```
gcc -dumpspecs | \
perl -p -e 's@/tools/lib/ld@/lib/ld@g;' \
-e 's@/tools/lib64/ld@/lib64/ld@g;' \
-e 's@\\*startfile_prefix_spec:\\n@$_/usr/lib/ @g;' > \
$(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 en fin de compte.

Remarquez que `/lib` ou bien `/lib64` est maintenant le préfixe de notre éditeur de liens dynamique.



Attention

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 prévu. Pour cela, effectuez une vérification d'intégrité :

Pour l'ABI 32 bits :

```
echo 'main(){}' > dummy.c
gcc ${BUILD32} dummy.c
readelf -l a.out | grep ': /lib'
```

Si tout fonctionne correctement, il ne devrait pas y avoir d'erreurs 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]
```

Pour l'ABI 64 bits :

```
echo 'main(){}' > dummy.c
gcc ${BUILD64} dummy.c
readelf -l a.out | grep ': /lib'
```

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

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

Remarquez que `/lib` ou bien `/lib64` est maintenant le préfixe de notre éditeur de liens dynamique.

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 de tests :

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


10.10. GMP-5.0.5 32 Bit Libraries

GMP est une bibliothèque pour faire de l'arithmétique en précision arbitraire sur les entiers, les nombres rationnels et les nombres flottants.

10.10.1. Installation de GMP



Remarque

Si vous compilez ce paquet sur un processeur différent de celui sur lequel vous envisagez d'exécuter le système CLFS, vous devez remplacer les enveloppes `config.guess` et `config.sub` de GMP par celles d'origine. Cela empêchera GMP de s'optimiser pour le mauvais processeur. Vous pouvez faire cette modification avec la commande suivante :

```
mv -v config{fsf,}.guess
mv -v config{fsf,}.sub
```

Préparez la compilation de GMP :

```
CPPFLAGS=-fexceptions CC="gcc -isystem /usr/include ${BUILD32}" \
CXX="g++ -isystem /usr/include ${BUILD32}" \
LDFLAGS="-Wl,-rpath-link,/usr/lib:/lib ${BUILD32}" \
ABI=32 ./configure --prefix=/usr \
--enable-cxx --enable-mpbsd
```

Compilez le paquet :

```
make
```



Important

La suite de tests pour GMP est considérée comme critique. Ne la sautez sous aucun prétexte.

Testez les résultats :

```
make check
```

Installez le paquet :

```
make install
```

L'entête installé par GMP est spécifique à l'architecture. Les programmes compilés en 32 bits demanderont l'entête fourni par l'installation 32 bits de GMP. De même pour les programmes 64 bits. Déplacez l'entête pour que l'enveloppe puisse le mettre à sa place plus tard:

```
mv -v /usr/include/gmp{,-32}.h
```

Les détails sur ce paquet sont disponibles dans Section 10.11.2, « Contenu de GMP. »

10.11. GMP-5.0.5 64 Bit

GMP est une bibliothèque pour faire de l'arithmétique en précision arbitraire sur les entiers, les nombres rationnels et les nombres flottants.

10.11.1. Installation de GMP



Remarque

Si vous compilez ce paquet sur un processeur différent de celui sur lequel vous envisagez d'exécuter le système CLFS, vous devez remplacer les enveloppes `config.guess` et `config.sub` de GMP par celles d'origine. Cela empêchera GMP de s'optimiser pour le mauvais processeur. Vous pouvez faire cette modification avec la commande suivante :

```
mv -v config{fsf,}.guess
mv -v config{fsf,}.sub
```

Préparez la compilation de GMP :

```
CPPFLAGS=-fexceptions CC="gcc -isystem /usr/include ${BUILD64}" \
CXX="g++ -isystem /usr/include ${BUILD64}" \
LDFLAGS="-Wl,-rpath-link,/usr/lib64:/lib64 ${BUILD64}" \
./configure --prefix=/usr \
--libdir=/usr/lib64 --enable-cxx --enable-mpbsd
```

Compilez le paquet :

```
make
```



Important

La suite de tests pour GMP est considérée comme critique. Ne la sautez sous aucun prétexte.

Testez les résultats :

```
make check
```

Installez le paquet :

```
make install
```

Créez du fichier d'entêtes 64 bits :

```
mv -v /usr/include/gmp{,-64}.h
```

Finalement, créez un morceau d'entête à la place des originaux:

```
cat > /usr/include/gmp.h << "EOF"
/* gmp.h - Stub Header */
#ifndef __STUB__GMP_H__
#define __STUB__GMP_H__

#if defined(__x86_64__) || \
    defined(__sparc64__) || \
    defined(__arch64__) || \
    defined(__powerpc64__) || \
    defined (__s390x__)
# include "gmp-64.h"
#else
# include "gmp-32.h"
#endif

#endif /* __STUB__GMP_H__ */
EOF
```

10.11.2. Contenu de GMP

Bibliothèques installées: libgmp.[a,so], libgmpxx.[a,so], libmp.[a,so]

Descriptions courtes

libgmp	Contient les définitions pour les fonctions GNU à précision multiple.
libgmpxx	Contient un emballer de classe C++ pour des types GMP.
libmp	Contient la bibliothèque de compatibilité Berkeley MP.

10.12. MPFR-3.1.1 32 Bit Libraries

La bibliothèque MPFR est une bibliothèque C pour des calculs de nombres flottants à précision multiple avec un arrondis correct.

10.12.1. Installation de MPFR

Préparez la compilation de MPFR :

```
CC="gcc -isystem /usr/include ${BUILD32}" \  
LDFLAGS="-Wl,-rpath-link,/usr/lib:/lib ${BUILD32}" \  
./configure --prefix=/usr --host=${CLFS_TARGET32} --enable-shared
```

Compilez le paquet :

```
make
```



Important

La suite de tests de MPFR est considéré comme critique. Ne la sautez sous aucun prétexte.

Testez les résultats :

```
make check
```

Installez les résultats :

```
make install
```

Les détails sur ce paquet sont disponibles dans Section 10.13.2, « Contenu de MPFR. »

10.13. MPFR-3.1.1 64 Bit

La bibliothèque MPFR est une bibliothèque C pour des calculs de nombres flottants à précision multiple avec un arrondis correct.

10.13.1. Installation de MPFR

Préparez la compilation de MPFR :

```
CC="gcc -isystem /usr/include ${BUILD64}" \
LD_FLAGS="-Wl,-rpath-link,/usr/lib64:/lib64 ${BUILD64}" \
./configure --prefix=/usr --libdir=/usr/lib64 --enable-shared
```

Compilez le paquet :

```
make
```



Important

La suite de tests de MPFR est considéré comme critique. Ne la sautez sous aucun prétexte.

Testez les résultats :

```
make check
```

Installez les résultats :

```
make install
```

10.13.2. Contenu de MPFR

Bibliothèques installées: libmpfr.[a,so]
Répertoire installé: /usr/share/doc/mpfr

Descriptions courtes

`libmpfr` La bibliothèque de nombres flojtants à précision multiple.

10.14. MPC-1.0.1 32 Bit Libraries

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

10.14.1. Installation de MPC

Préparez la compilation de MPC :

```
CC="gcc -isystem /usr/include ${BUILD32}" \  
LDFLAGS="-Wl,-rpath-link,/usr/lib:/lib ${BUILD32}" \  
./configure --prefix=/usr --host=${CLFS_TARGET32}
```

Compilez le paquet :

```
make
```



Important

La suite de tests de MPC est considérée comme critique. Ne la sautez en aucune circonstance.

Testez les résultats :

```
make check
```

Installez le paquet :

```
make install
```

Les détails sur ce paquet sont disponibles dans Section 10.15.2, « Contenu de MPC. »

10.15. MPC-1.0.1 64 Bit

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

10.15.1. Installation de MPC

Préparez la compilation de MPC :

```
CC="gcc -isystem /usr/include ${BUILD64}" \
LDFLAGS="-Wl,-rpath-link,/usr/lib64:/lib64 ${BUILD64}" \
./configure --prefix=/usr --libdir=/usr/lib64
```

Compilez le paquet :

```
make
```



Important

La suite de tests de MPC est considérée comme critique. Ne la sautez en aucune circonstance.

Testez les résultats :

```
make check
```

Installez le paquet :

```
make install
```

10.15.2. Contenu de MPC

Bibliothèques installées: libmpc.[a,so]

Descriptions courtes

libmpc La bibliothèque Multiple Precision Complex.

10.16. PPL-0.12.1 32 Bit Libraries

La bibliothèque *Parma Polyhedra Library* (PPL) fournit des abstractions numériques destinées principalement à des applications dans le domaine de l'analyse et de la vérification de systèmes complexes. CLooG-PPL exige cette bibliothèque.

10.16.1. Installation de PPL

Préparez la compilation de PPL :

```
CPPFLAGS=-fexceptions CC="gcc -isystem /usr/include ${BUILD32}" \
CXX="g++ -isystem /usr/include ${BUILD32}" \
LDFLAGS="-Wl,-rpath-link,/usr/lib:/lib ${BUILD32}" \
./configure --prefix=/usr --host=${CLFS_TARGET32} \
--enable-shared --disable-optimization
```

Compilez le paquet :

```
make
```



Important

La suite de tests de PPL est considérée comme critique. Ne la sautez sous aucun prétexte.

Testez les résultats :

```
make check
```

Installez le paquet :

```
make install
```

Préparez `ppl-config` à être enveloppé par l'enveloppeur multiarchitecture et ensuite l'enveloppe :

```
mv -v /usr/bin/ppl-config{,-32}
```

Une des entêtes installées par PPL est spécifique à l'architecture. Les programmes compilés en 32 bits demanderont les entêtes fournies par l'installation 32 bits de PPL. Les mêmes choses pour les programmes 64 bits. Déplacez l'en-tête de sorte qu'une enveloppe puisse être mise à sa place plus tard :

```
mv -v /usr/include/ppl{,-32}.hh
```

Les détails sur ce paquet sont disponibles dans Section 10.17.2, « Contenu de PPL. »

10.17. PPL-0.12.1 64 Bit

La bibliothèque *Parma Polyhedra Library* (PPL) fournit des abstractions numériques destinées principalement à des applications dans le domaine de l'analyse et de la vérification de systèmes complexes. CLooG-PPL exige cette bibliothèque.

10.17.1. Installation de PPL

Préparez la compilation de PPL :

```
CPPFLAGS=-fexceptions CC="gcc -isystem /usr/include ${BUILD64}" \
CXX="g++ -isystem /usr/include ${BUILD64}" \
LDFLAGS="-Wl,-rpath-link,/usr/lib64:/lib64 ${BUILD64}" \
./configure --prefix=/usr --libdir=/usr/lib64 \
--enable-shared --disable-optimization
```

Compilez le paquet :

```
make
```



Important

La suite de tests de PPL est considérée comme critique. Ne la sautez sous aucun prétexte.

Testez les résultats :

```
make check
```

Installez le paquet :

```
make install
```

Préparez `ppl-config` à être enveloppé par l'enveloppeur multiarchitecture et ensuite l'enveloppe :

```
mv -v /usr/bin/ppl-config{,-64}
ln -svf multiarch_wrapper /usr/bin/ppl-config
```

Crée le fichier d'en-tête 64 bits :

```
mv -v /usr/include/ppl{,-64}.hh
```

Finalement, crée un morceau d'entête à la place des originaux:

```
cat > /usr/include/ppl.hh << "EOF"
/* ppl.hh - Stub Header */
#ifndef __STUB__PPL_HH__
#define __STUB__PPL_HH__

#if defined(__x86_64__) || \
    defined(__sparc64__) || \
    defined(__arch64__) || \
    defined(__powerpc64__) || \
    defined (__s390x__)
# include "ppl-64.h"
#else
# include "ppl-32.h"
#endif

#endif /* __STUB__PPL_HH__ */
EOF
```

10.17.2. Contenu de PPL

Programmes installés: ppl-config, ppl_lcdd, ppl_pips
Bibliothèques installées: libppl.[a,so], libppl_c.[a,so]
Répertoires installés: /usr/share/doc/ppl

Descriptions courtes

ppl-config	Affiche des informations sur l'installation de PPL
ppl_lcdd	Lit une représentation H d'un polyèdre et génère une représentation V du même polyèdre
ppl_pips	Un résolveur de problèmes de programmation paramétrique d'entiers basé sur PPL
libppl	La bibliothèque <i>Parma Polyhedra Library</i> (PPL).
libppl_c	Les bindings de Parma Polyhedra Library pour C.
libpwl	La bibliothèque Parma Watchdog Library

10.18. CLooG-0.16.3 32 Bit Libraries

CLooG est une bibliothèque pour générer du code pour analyser des polyèdres Z. En d'autres termes, il trouve du code qui atteint chaque point entier (ou intégral) d'un ou plusieurs polyèdres paramétrés. GCC se lie à cette bibliothèque afin d'activer le nouveau code de génération de boucle, connu en tant que Graphite.

10.18.1. Installation de CLooG

Préparez la compilation de CLooG :

```
CC="gcc -isystem /usr/include ${BUILD32}" \
LDFLAGS="-Wl,-rpath-link,/usr/lib:/lib ${BUILD32}" \
./configure --prefix=/usr \
--host=${CLFS_TARGET32} --enable-shared
```

Compilez le paquet :

```
make
```



Important

La suite de tests de CLooG est considérée comme critique. Ne la sautez sous aucun prétexte.

Testez les résultats :

```
make check
```

Installez le paquet :

```
make install
```

Les détails sur ce paquet sont disponibles dans Section 10.19.2, « Contenu de CLooG. »

10.19. CLooG-0.16.3 64 Bit

CLooG est une bibliothèque pour générer du code pour analyser des polyèdres Z. En d'autres termes, il trouve du code qui atteint chaque point entier (ou intégral) d'un ou plusieurs polyèdres paramétrés. GCC se lie à cette bibliothèque afin d'activer le nouveau code de génération de boucle, connu en tant que Graphite.

10.19.1. Installation de CLooG

Préparez la compilation de CLooG :

```
CC="gcc -isystem /usr/include ${BUILD64}" \
LDLAGS="-Wl,-rpath-link,/usr/lib64:/lib64 ${BUILD64}" \
./configure --prefix=/usr \
--libdir=/usr/lib64 --enable-shared
```

Compilez le paquet :

```
make
```



Important

La suite de tests de CLooG est considérée comme critique. Ne la sautez sous aucun prétexte.

Testez les résultats :

```
make check
```

Installez le paquet :

```
make install
```

10.19.2. Contenu de CLooG

Programme installé:	cloog
Bibliothèques installées:	libcloog-isl.[a,so], libisl.[a,so]
Répertoires installés:	/usr/include/cloog, /usr/include/isl

Descriptions courtes

cloog	Générateur de boucle pour l'analyse de polyèdres Z
libcloog-isl	Fondation isl pour CLooG.
libisl	La bibliothèque Integer Set.

10.20. Zlib-1.2.7 32 Bit Libraries

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

10.20.1. Installation de Zlib

Préparez la compilation de Zlib :

```
CC="gcc -isystem /usr/include ${BUILD32}" \  
CXX="g++ -isystem /usr/include ${BUILD32}" \  
LDFLAGS="-Wl,-rpath-link,/usr/lib:/lib ${BUILD32}" \  
./configure --prefix=/usr
```

Compilez le paquet :

```
make
```

Pour tester les résultats, exécutez : **make check**.

Installez le paquet :

```
make install
```

La commande a installé deux fichiers `.so` dans `/usr/lib`. Nous allons les déplacer dans `/lib` puis le lier à nouveau à `/usr/lib` :

```
mv -v /usr/lib/libz.so.* /lib  
ln -svf ../../lib/libz.so.1 /usr/lib/libz.so
```

Les détails sur ce paquet sont disponibles dans Section 10.21.2, « Contenu de Zlib. »

10.21. Zlib-1.2.7 64 Bit

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

10.21.1. Installation de Zlib

Préparez la compilation de Zlib :

```
CC="gcc -isystem /usr/include ${BUILD64}" \
CXX="g++ -isystem /usr/include ${BUILD64}" \
LDFLAGS="-Wl,-rpath-link,/usr/lib64:/lib64 ${BUILD64}" \
./configure --prefix=/usr --libdir=/usr/lib64
```

Compilez le paquet :

```
make
```

Pour tester les résultats, exécutez : **make check**.

Installez le paquet :

```
make install
```

La commande précédente a installé deux fichiers `.so` dans `/usr/lib64`. Nous allons les déplacer dans `/lib64` et ensuite les relier à `to /usr/lib64`:

```
mv -v /usr/lib64/libz.so.* /lib64
ln -svf ../../lib64/libz.so.1 /usr/lib64/libz.so
```

10.21.2. Contenu de Zlib

Bibliothèques installées: `libz.[a,so]`

Descriptions courtes

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

10.22. Binutils-2.23

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

10.22.1. Installation de Binutils

Vérifiez que les pseudo-terminaux (PTY) fonctionnent correctement dans l'environnement chroot. Vérifiez que tout est bien configuré en effectuant un simple test :

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

Cette commande devrait donner la sortie suivante :

```
spawn ls
```

Si, à la place, elle donne un message disant qu'il faut créer plus de pty, alors l'environnement n'est pas bien paramétré pour l'opération PTY. Ce problème doit être résolu avant de lancer les suites de tests pour Binutils et GCC.

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 :

```
CC="gcc -isystem /usr/include ${BUILD64}" \
LDFlags="-Wl,-rpath-link,/usr/lib64:/lib64:/usr/lib:/lib ${BUILD64}" \
  ../binutils-2.23/configure --prefix=/usr \
  --enable-shared --enable-64-bit-bfd --libdir=/usr/lib64
```

Compilez le paquet :

```
make configure-host
```



Important

Pendant **make configure-host** il se peut que vous receviez le message d'erreur suivant. Vous pouvez l'ignorer en toute sécurité.

```
WARNING: `flex' is missing on your system. You should only
need it if you modified a `.l' file. You may need the `Flex'
package in order for those modifications to take effect. You
can get `Flex' from any GNU archive site.
```

```
make tooldir=/usr
```

Voici la signification du paramètre de make :

```
tooldir=/usr
```

Normalement, le répertoire `tooldir` (celui où seront placés les exécutables) est configuré avec `$(exec_prefix)/$(target_alias)`. Comme il s'agit d'un système personnalisé, nous n'avons pas besoin d'un répertoire spécifique à notre cible dans `/usr`.

**Important**

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

Testez les résultats :

```
make check
```

Installez le paquet :

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

Installez le fichier d'en-tête `libiberty` dont ont besoin certains paquets :

```
cp -v ../binutils-2.23/include/libiberty.h /usr/include
```

10.22.2. Contenu de Binutils

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

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 des archives
as	Un assembleur qui assemble la sortie de gcc en des fichiers objets
c++filt	Utilisé par l'éditeur de liens pour récupérer les symboles C++ et Java, et pour empêcher les fonctions surchargées d'arrêter brutalement le programme
elfedit	Mise à jour de 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 en 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 (<i>Binary File Descriptor</i>)
libopcodes	Une bibliothèque de gestion des opcodes—la « version lisible » des instructions du processeur ; elle est utilisée pour construire des outils comme objdump .

10.23. GCC-4.6.3

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

10.23.1. Installation de GCC

Le correctif suivant contient un certain nombre de mises à jour vers la branche 4.6.3, faites par les développeurs de GCC :

```
patch -Np1 -i ../gcc-4.6.3-branch_update-2.patch
```

Appliquez une substitution **sed** qui va supprimer l'installation de `libiberty.a`. La version of `libiberty.a` fournie par Binutils sera utilisée à la place :

```
sed -i 's/install_to_${INSTALL_DEST} //' libiberty/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 :

```
CC="gcc -isystem /usr/include ${BUILD64}" \
CXX="g++ -isystem /usr/include ${BUILD64}" \
LDFLAGS="-Wl,-rpath-link,/usr/lib64:/lib64:/usr/lib:/lib" \
../gcc-4.6.3/configure --prefix=/usr --libdir=/usr/lib64 \
--libexecdir=/usr/lib64 --enable-shared --enable-threads=posix \
--enable-__cxa_atexit --enable-c99 --enable-long-long \
--enable-clocale=gnu --enable-languages=c,c++ --disable-libstdcxx-pch \
--enable-cloog-backend=isl
```

Compilez le paquet :

```
make
```



Important

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

Testez les résultats mais ne vous arrêtez pas aux erreurs :

```
make -k check
```

L'option `-k` est utilisé pour que la suite de test s'exécute jusqu'à la fin et ne s'arrête pas au premier échec. La suite de tests de GCC est très complète et il est presque certain qu'elle générera quelques échecs. Pour recevoir un résumé des résultats de la suite de tests, lancez :

```
../gcc-4.6.3/contrib/test_summary
```

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

Quelques échecs inattendus sont inévitables. Les développeurs de GCC connaissent ces problèmes, mais ne les ont pas encore résolus.

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
```

10.23.2. Contenu de GCC

Programmes installés: `c++`, `cc` (link to `gcc`), `cpp`, `g++`, `gcc` et `gcov`
Bibliothèques installées: `libgcc.a`, `libgcc_eh.a`, `libgcc_s.so`, `libgcov.a`, `libgomp.[a,so]`, `libmudflap.[a,so]`, `libmudflapth.[a,so]`, `libssp.[a,so]`, `libssp_nonshared.a`, `libstdc++.a`, `libstdc++.a` et `libsupc++`
Répertoires installés: `/usr/include/c++`, `/usr/lib/gcc`, `/usr/share/gcc-4.6.3`

Descriptions courtes

cc	Le compilateur C
cpp	Le préprocesseur C ; il est utilisé par le compilateur pour l'extension des instructions <code>#include</code> , <code>#define</code> et d'autres instructions similaires dans les fichiers sources
c++	Le compilateur C++
g++	Le compilateur C++
gcc	Le compilateur C
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	Bibliothèque qui est liée à un programme lorsqu'on demande à gcc d'activer le profilage
libgomp	une implémentation GNU de l'API OpenMP pour la programmation en C/C++ et Fortran de mémoire parallèle partagée pour plusieurs plateformes
libmudflap	Les bibliothèques <code>libmudflap</code> sont utilisées par GCC pour des opérations d'instrumentalisation des pointeurs et de déréférencement des tableaux.
libssp	Contient le support des routines pour la fonctionnalité de protecteur Stack-mashing de GCC
libstdc++	La bibliothèque C++ standard
libsupc++	Fournit des routines pour le support du langage de programmation C++

10.24. Création d'un programme enveloppe multi-architecture (« Multiarch Wrapper »)

Le programme enveloppe multi-architecture, ou « Multiarch Wrapper », sert à envelopper certains binaires contenant en dur les chemins vers leurs bibliothèques ou bien qui sont spécifiques à certaines architectures..

10.24.1. Installation de The Multiarch Wrapper

```

cat > multiarch_wrapper.c << "EOF"
#define _GNU_SOURCE

#include <errno.h>
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <unistd.h>

#ifndef DEF_SUFFIX
# define DEF_SUFFIX "64"
#endif

int main(int argc, char **argv)
{
    char *filename;
    char *suffix;

    if(!(suffix = getenv("USE_ARCH")))
        if(!(suffix = getenv("BUILDENV")))
            suffix = DEF_SUFFIX;

    if (asprintf(&filename, "%s-%s", argv[0], suffix) < 0) {
        perror(argv[0]);
        return -1;
    }

    int status = EXIT_FAILURE;
    pid_t pid = fork();

    if (pid == 0) {
        execvp(filename, argv);
        perror(filename);
    } else if (pid < 0) {
        perror(argv[0]);
    } else {
        if (waitpid(pid, &status, 0) != pid) {
            status = EXIT_FAILURE;
            perror(argv[0]);
        } else {
            status = WEXITSTATUS(status);
        }
    }
    free(filename);

    return status;
}

EOF

```

Compilez et installez le « Multiarch Wrapper » :

```
gcc ${BUILD64} multiarch_wrapper.c -o /usr/bin/multiarch_wrapper
```

Ce programme sera utilisé plus tard dans le livre avec Perl. Il sera également très utile hors du système CLFS de base.

Faisons un test :

```
echo 'echo "Version 32 bits"' > test-32
echo 'echo "Version 64 bits"' > test-64
chmod -v 755 test-32 test-64
ln -sv /usr/bin/multiarch_wrapper test
```

Testons le programme enveloppe :

```
USE_ARCH=32 ./test
USE_ARCH=64 ./test
```

Le résultat de ces commandes devrait être alors :

```
32bit Version
64bit Version
```

Supprimez les sources, les binaires et le lien testcase :

```
rm -v multiarch_wrapper.c test{-32,-64}
```

10.24.2. Contenu du Multiarch Wrapper

programmes installés: multiarch_wrapper

Descriptions courtes

multiarch_wrapper Exécute un programme différent en se basant sur la variable `USE_ARCH`. Cette variable `USE_ARCH` sera alors le suffixe du programme exécuté.

10.25. Sed-4.2.1

Le paquet Sed contient un éditeur de flux.

10.25.1. Installation de Sed

Préparez la compilation de Sed :

```
CC="gcc ${BUILD64}" ./configure --prefix=/usr \  
--bindir=/bin
```

Compilez le paquet :

```
make
```

Compilez 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
```

10.25.2. Contenu de Sed

Programme installé:	sed
Répertoire installé:	/usr/share/doc/sed

Descriptions courtes

sed Filtre et transforme des fichiers texte en une seule passe

10.26. Bibliothèques Ncurses-5.9 32 bits

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

10.26.1. Installation de Ncurses

Le correctif suivant incorpore les mises à jour de la branche 5.9 issue des développeurs de Ncurses :

```
patch -Np1 -i ../ncurses-5.9-branch_update-4.patch
```

Préparez la compilation de Ncurses :

```
CC="gcc ${BUILD32}" CXX="g++ ${BUILD32}" \
./configure --prefix=/usr --libdir=/lib \
--with-shared --without-debug --enable-widec \
--with-manpage-format=normal \
--with-default-terminfo-dir=/usr/share/terminfo
```

Compilez le paquet :

```
make
```

Ce paquet a une suite de tests qu'on peut lancer après avoir installé le paquet. Les tests se trouvent dans le répertoire `test/`. Voir le fichier `README` de ce répertoire pour les détails.

Installez le paquet :

```
make install
```

Préparez `ncursesw5-config` à être emballé par le multiarch wrapper :

```
mv -v /usr/bin/ncursesw5-config{,-32}
```

Déplacez les bibliothèques statiques de Ncurses au bon endroit :

```
mv -v /lib/lib{panelw,menuw,formw,ncursesw,ncurses++w}.a /usr/lib
```

Créez des liens symboliques dans `/usr/lib`:

```
rm -v /lib/lib{ncursesw,menuw,panelw,formw}.so
ln -svf ../../lib/libncursesw.so.5 /usr/lib/libncursesw.so
ln -svf ../../lib/libmenuw.so.5 /usr/lib/libmenuw.so
ln -svf ../../lib/libpanelw.so.5 /usr/lib/libpanelw.so
ln -svf ../../lib/libformw.so.5 /usr/lib/libformw.so
```

Maintenant, nous allons rendre notre Ncurses compatible pour que les vieux programmes non compatibles avec `widec` se construisent correctement :

```
for lib in curses ncurses form panel menu ; do
    echo "INPUT(-l${lib}w)" > /usr/lib/lib${lib}.so
    ln -sfv lib${lib}w.a /usr/lib/lib${lib}.a
done
ln -sfv libcurses.so /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
ln -sfv libncurses++w.a /usr/lib/libncurses++.a
ln -sfv ncursesw5-config-32 /usr/bin/ncurses5-config-32
```


Maintenant, nous allons créer un lien symbolique pour `/usr/share/terminfo` dans `/usr/lib` pour des questions de compatibilité :

```
ln -sfv ../share/terminfo /usr/lib/terminfo
```

Les détails sur ce paquet sont disponibles dans Section 10.27.2, « Contenu de Ncurses. »

10.27. Ncurses-5.9 64 bits

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

10.27.1. Installation de Ncurses

Le correctif suivant incorpore les mises à jour de la branche 5.9 issue des développeurs de Ncurses :

```
patch -Np1 -i ../ncurses-5.9-branch_update-4.patch
```

Préparez la compilation de Ncurses :

```
CC="gcc ${BUILD64}" CXX="g++ ${BUILD64}" \
./configure --prefix=/usr --libdir=/lib64 \
--with-shared --without-debug --enable-widec \
--with-manpage-format=normal \
--with-default-terminfo-dir=/usr/share/terminfo
```

Compilez le paquet :

```
make
```

Ce paquet a une suite de tests qu'on peut lancer après avoir installé le paquet. Les tests se trouvent dans le répertoire `test/`. Voir le fichier `README` de ce répertoire pour les détails.

Installez le paquet :

```
make install
```

Préparez `ncursesw5-config` à être emballé par le multiarch wrapper puis emballez-le via un lien vers ce dernier :

```
mv -v /usr/bin/ncursesw5-config{,-64}
ln -svf multiarch_wrapper /usr/bin/ncursesw5-config
```

Déplacez les bibliothèques statiques de Ncurses au bon endroit :

```
mv -v /lib64/lib{panelw,menuw,formw,ncursesw,ncurses++w}.a /usr/lib64
```

Créez les liens symboliques dans `/usr/lib64` :

```
rm -v /lib64/lib{ncursesw,menuw,panelw,formw}.so
ln -svf ../../lib64/libncursesw.so.5 /usr/lib64/libncursesw.so
ln -svf ../../lib64/libmenuw.so.5 /usr/lib64/libmenuw.so
ln -svf ../../lib64/libpanelw.so.5 /usr/lib64/libpanelw.so
ln -svf ../../lib64/libformw.so.5 /usr/lib64/libformw.so
```

Maintenant, nous allons rendre notre Ncurses compatible pour que les vieux programmes non compatibles avec widec se construisent correctement :

```
for lib in curses ncurses form panel menu ; do
    echo "INPUT(-l${lib}w)" > /usr/lib64/lib${lib}.so
    ln -sfv lib${lib}w.a /usr/lib64/lib${lib}.a
done
ln -sfv libcurses.so /usr/lib64/libcursesw.so
ln -sfv libncurses.so /usr/lib64/libcurses.so
ln -sfv libncursesw.a /usr/lib64/libcursesw.a
ln -sfv libncurses.a /usr/lib64/libcurses.a
ln -sfv libncurses++w.a /usr/lib64/libncurses++.a
ln -sfv ncursesw5-config-64 /usr/bin/ncurses5-config-64
ln -sfv ncursesw5-config /usr/bin/ncurses5-config
```

Puis nous créerons un lien symbolique vers /usr/share/terminfo dans /usr/lib64 pour préserver la compatibilité :

```
ln -sfv ../share/terminfo /usr/lib64/terminfo
```

10.27.2. Contenu de Ncurses

Programmes installés:	captinfo (lien vers tic), clear, infocmp, infotocap (lien vers tic), ncursesw5-config, reset (lien vers tset), tabs, tic, toe, tput et tset
Bibliothèques installées:	libcursesw.so (link to libncursesw.so), libformw.[a,so] et libpanelw.[a,so]
Répertoires installés:	/usr/share/tabset, /usr/share/terminfo

Descriptions courtes

captinfo	Convertit une description termcap en description terminfo
clear	Vide l'écran, si possible
infocmp	Compare ou affiche des descriptions terminfo
infotocap	Convertit une description terminfo en description termcap
ncursesw5-config	Fournit des informations de configuration de ncurses
reset	Réinitialise un terminal à ses valeurs par défaut
tabs	Initialise et vide des taquets de tab sur un terminal
tic	Le compilateur entrée-description qui traduit un fichier terminfo à partir du format source en format binaire requis pour les routines de la bibliothèque ncurses. Un fichier terminfo contient des informations sur les possibilités d'un terminal donné
toe	Liste tous les types de terminaux disponibles, en donnant leur nom primaire et leur description
tput	Rend disponibles les fonctionnalités dépendantes du terminal pour le shell ; il peut aussi être utilisé pour réinitialiser ou paramétrer un terminal ou d'afficher son nom long
tset	Peut être utilisé pour initialiser des terminaux
libcursesw	Un lien vers libncursesw
libncursesw	Contient des fonctions pour afficher du texte de façons complexes sur un écran de terminal ; un bon exemple d'utilisation de ces fonctions est le menu affiché par le make menuconfig du noyau
libformw	Contient des fonctions pour implémenter des formulaires

<code>libmenuw</code>	Contient des fonctions pour implémenter des menus
<code>libpanelw</code>	Contient des fonctions pour implémenter des panneaux

10.28. Pkg-config-lite-0.27.1-1

Pkg-config est un outil pour vous aider à insérer les bonnes options du compilateur sur la ligne de commande lors de la compilation d'applications et de bibliothèques.

10.28.1. Installation de Pkg-config

Préparez la compilation de Pkg-config :

```
CC="gcc ${BUILD64}" ./configure --prefix=/usr \
  --with-pc-path=/usr/share/pkgconfig
```

Compilez le paquet :

```
make
```

Pour tester les résultats, effectuez : **make check**.

Installez le paquet :

```
make install
```

Sur des constructions multilib, le répertoire de la bibliothèque a été enlevé du path de recherche par défaut de **pkg-config**. Réglez des variables d'environnement pour aider à régler correctement le chemin à l'avenir :

```
export PKG_CONFIG_PATH32="/usr/lib/pkgconfig"
export PKG_CONFIG_PATH64="/usr/lib64/pkgconfig"
```

Exportez ces variables pour empêcher toutes choses dans le futur.

```
cat >> /root/.bash_profile << EOF
export PKG_CONFIG_PATH32="${PKG_CONFIG_PATH32}"
export PKG_CONFIG_PATH64="${PKG_CONFIG_PATH64}"
EOF
```

10.28.2. Contenu de Pkg-config

Programmes installés: pkg-config
Répertoire installé: /usr/share/doc/pkg-config

Descriptions courtes

pkg-config Le programme **pkg-config** est utilisé pour récupérer des informations sur les bibliothèques installées dans le système. On l'utilise en général pour compiler et lier à une ou plusieurs bibliothèques.

10.29. Util-linux-2.22.1 32 bits

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.

10.29.1. Installation de Util-linux

Préparez la compilation d'Util-linux :

```
CC="gcc ${BUILD32}" ./configure --libdir=/lib \
  --enable-arch --enable-write --disable-login --disable-su
```

Voici la signification des options de configure :

--disable-login --disable-su

Désactive la construction des programmes **login** et **su** puisque le paquet Shadow installe ses propres versions.

--enable-write

Cette option permet au programme **write** d'être installé.

Compilez le paquet :

```
make
```

Pour tester les résultats, lancez : **make check**.

Installez le paquet :

```
make install
```

Les détails sur ce paquet sont disponibles dans Section 10.30.3, « Contenu de Util-linux. »

10.30. Util-linux-2.22.1 64 bits

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.

10.30.1. 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' .)
    hwclock/hwclock.c
mkdir -pv /var/lib/hwclock
```

10.30.2. Installation de Util-linux

Préparez la compilation d'Util-linux :

```
CC="gcc ${BUILD64}" ./configure --libdir=/lib64 \
    --enable-arch --enable-write --disable-login --disable-su
```

Voici la signification des options de configure :

`--disable-login --disable-su`

Désactive la construction des programmes **login** et **su** puisque le paquet Shadow installe ses propres versions.

`--enable-write`

Cette option permet au programme **write** d'être installé.

Compilez le paquet :

```
make
```

Pour tester les résultats, lancez : **make check**.

Installez le paquet :

```
make install
```

Déplacez le binaire **logger** vers `/bin` selon le besoin du paquet CLFS-Bootscripts :

```
mv -v /usr/bin/logger /bin
```

10.30.3. Contenu de Util-linux

Programmes installés: addpart,agetty,arch,blkid,blockdev,cal,cfdisk,chequ,chrt,col,colcrt,colrm,column,ctrlaltdel,cytune,delpart,dmesg,eject,fallocate,fdformat,fdisk,findfs,findmnt,flock,fsck,fsck.cramfs,fsck.minix,fsfreeze,fstrim,getopt,hexdump,hwclock,ionice,ipcmk,ipcrm,ipcs,isosize,kill,ldattach,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,sulogin,swapon,swapoff (lien vers swapon),swapon,switch_root,tailf,taskset,tunelp,ul,umount,unshare,utmpdump,uidd,uuidgen,wall,wddctl,whereis,wipefs,et write

Bibliothèques installées: libblkid.[a,so], libmount.[a,so] et 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 de nouvelles partitions
agetty	Ouvre un port tty, demande un nom de connexion puis appelle le programme login
arch	Signale l'architecture de la machine
blkid	A command line utility to locate and print block device attributes
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	Outil pour configurer les 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
ddate	Donne la date discordienne ou convertit la date grégorienne en une date discordienne
delpart	Demande au noyau de supprimer une partition
dmesg	Affiche les messages du noyau lors du démarrage
eject	Éjecte un média amovible
fallocate	Pré-affecte de la place pour 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	Finds a file system by label or Universally Unique Identifier (UUID)
findmnt	Liste les systèmes de fichiers montés ou recherche un système de fichiers
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	Suspend et rétablit l'accès au système de fichiers
fstrim	Désactive les blocs non utilisés d'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 (<i>Real-Time Clock</i> , horloge à temps réel) ou horloge BIOS (<i>Basic Input-Output System</i>)

ionice	Donne 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
isozsize	Affiche la taille d'un système de fichiers iso9660
kill	Envoie un signal à un processus
ldattach	Attache une ligne de discipline à une ligne en série.
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	Affiche des informations sur les périphériques de bloc
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	Vous dit si un répertoire est ou pas 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 définit des limites de ressources pour un processus
raw	Associe un périphérique de caractère Linux raw à 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	Met un système en sommeil jusqu'à un moment de réveil spécifié
script	Crée une transcription de session du terminal à partir de tout ce qui est affiché sur un terminal
scriptreplay	Rejoue une transcription créée par script

setarch	Change d'architecture signalée dans un nouvel environnement de programme et initialise les commutateurs de personnalité
setsid	Lance le programme donné dans une nouvelle session
setterm	Initialise les attributs du terminal
sfdisk	Un manipulateur de table de partitions disque
sulogin	Permet à <i>root</i> de se connecter ; appelée normalement par init quand le système passe en mode mono-utilisateur
swapon	Active les périphériques et fichiers de pagination et d'échange, et liste les périphériques et fichiers en cours d'utilisation.
swapoff	Désactive les périphériques et fichiers de pagination et d'échange.
swlabel	Affiche ou modifie l'étiquette ou l'UUID d'une zone d'échange
switch_root	Change vers un autre système de fichiers pour racine de l'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 l'affinité processeur du processus
tunelp	Est utilisé pour paramétrer finement une imprimante ligne
ul	Un filtre pour traduire les soulignements en séquences d'échappement indiquant un soulignement pour le terminal utilisé
umount	Déconnecte un système de fichiers à partir de la hiérarchie de fichiers du système
unshare	Lance un programme avec certains espaces nommés non partagés avec celui parent
utmpdump	Affiche le contenu du fichier de connexion donné dans un format plus convivial
uudd	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	Écrit un message à tous les utilisateurs connectés
wdctl	Affiche l'état du watchdog matériel
whereis	Affiche l'emplacement du binaire, les sources et la page de manuel de la commande donnée
wipefs	Enlève d'un périphérique la signature d'un système de fichiers
write	Envoie un message à l'utilisateur donné <i>sauf si</i> l'utilisateur a désactivé de tels messages
libblkid	Contient des routines pour l'identification de processus et l'extraction de jetons
libmount	Contient des routines pour analyser les fichiers <i>/etc/fstab</i> , <i>/etc/mtab</i> et <i>/proc/self/mountinfo</i> , gérer <i>/etc/mtab</i> et configurer diverses options de montage
libuuid	Contient des routines pour générer des identifiants uniques pour les objets qui pourraient être accessibles en dehors du système local

10.31. Bibliothèques Procps-3.2.8 32 bits

Le paquet Procps contient des programmes pour surveiller les processus.

10.31.1. Installation de Procps

Le correctif suivant ajoute le support des groupes de contrôle des processus à ps :

```
patch -Np1 -i ../procps-3.2.8-ps_cgroup-1.patch
```

Le correctif suivant corrige un problème qui faisait que certains outils procps affichent une erreur à l'écran si le moniteur ne fonctionne pas à 60Hz :

```
patch -Np1 -i ../procps-3.2.8-fix_HZ_errors-1.patch
```

Ce qui suit corrige un problème avec Make 3.82 :

```
sed -i -r '/^include/s/\*(.*)/proc\1 ps\1/' Makefile
```

Compilez le paquet :

```
make CC="gcc ${BUILD32}" m64=""
```

Ce paquet ne fournit pas de suite de tests.

Installez le paquet :

```
make SKIP='/bin/kill /usr/share/man/man1/kill.1' install lib64=lib
```

Voici la signification des options d'installation et de celles de make :

```
CC="gcc ${BUILD32}"
```

Ceci permet de compiler en utilisant notre gcc avec les options contenues dans la variable `${BUILD32}`.

```
m64=""
```

Le fichier `Makefile` de ce paquet présuppose de compiler en 64 bits dès que cela est possible. Dans CLFS, nous compilons indépendamment pour chaque ABI disponible. Forcer le paramètre `m64` permet d'éviter ce comportement.

```
lib64=lib
```

Le fichier `Makefile` tente également d'installer dans `lib64` en environnement multilib. Une fois encore, nous évitons ce comportement.

Les détails sur ce paquet sont disponibles dans Section 10.32.2, « Contenu de Procps. »

10.32. Procps-3.2.8 64 bits

Le paquet Procps contient des programmes pour surveiller les processus.

10.32.1. Installation de Procps

Le correctif suivant ajoute le support des groupes de contrôle des processus à ps :

```
patch -Np1 -i ../procps-3.2.8-ps_cgroup-1.patch
```

Le correctif suivant corrige un problème qui faisait que certains outils procps affichent une erreur à l'écran si le moniteur ne fonctionne pas à 60Hz :

```
patch -Np1 -i ../procps-3.2.8-fix_HZ_errors-1.patch
```

Ce qui suit corrige un problème avec Make 3.82 :

```
sed -i -r '/^include/s/\*(.*)/proc\1 ps\1/' Makefile
```

Compilez le paquet :

```
make CC="gcc ${BUILD64}" m64=""
```

Ce paquet ne fournit pas de suite de tests.

Installez le paquet :

```
make SKIP='/bin/kill /usr/share/man/man1/kill.1' install lib64=lib64
```

Voici la signification des options d'installation :

```
CC="gcc ${BUILD64}"
```

Ceci permet de compiler en utilisant notre gcc avec les options contenues dans la variable `${BUILD64}`.

```
m64=""
```

Le fichier `Makefile` de ce paquet présume de compiler en 64 bits dès que cela est possible. Dans CLFS, nous compilons indépendamment pour chaque ABI disponible. Forcer le paramètre `m64` permet d'éviter ce comportement.

```
lib64=lib64
```

Le fichier `Makefile` tente également d'installer dans `lib64` en environnement multilib. Une fois encore, nous évitons ce comportement.

10.32.2. Contenu de Procps

Programmes installés:	free, pgrep, pkill, pmap, ps, pwdx, skill, slabtop, snice, sysctl, tload, top, uptime, vmstat, w et watch
Répertoire installé:	libproc.so

Descriptions courtes

free	Indique la quantité de mémoire libre et utilisée 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
skill	Envoie des signaux aux processus correspondant à un critère donné
slabtop	Affiche des informations détaillées sur le cache slab du noyau en temps réel
snice	Modifie les priorités des processus suivant le critère donné.
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 plus 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

10.33. Bibliothèques E2fsprogs-1.42.6 32 bits

Le paquet E2fsprogs contient les outils de gestion du système de fichiers `ext2`. Il supporte aussi les systèmes de fichiers journalisés `ext3` et `ext4`.

10.33.1. Installation de E2fsprogs

La documentation d'E2fsprogs recommande de construire le paquet dans un sous-répertoire du répertoire source :

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

Préparez la compilation d'E2fsprogs :

```
CC="gcc ${BUILD32}" PKG_CONFIG_PATH="${PKG_CONFIG_PATH32}" \
  ../configure --prefix=/usr --with-root-prefix="" \
    --enable-elf-shlibs --disable-libblkid \
    --disable-libuuid --disable-fsck \
    --disable-uuid
```

Voici la signification des options de configure :

--with-root-prefix=""

Certains programmes (comme **e2fsck**) sont considérés 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.

Compilez les bibliothèques :

```
make libs
```

Installez les bibliothèques statiques et les en-têtes :

```
make install-libs
```

Les détails sur ce paquet sont disponibles dans Section 10.34.2, « Contenu de E2fsprogs. »

10.34. E2fsprogs-1.42.6 64 bits

Le paquet E2fsprogs contient les outils de gestion du système de fichiers ext2. Il supporte aussi les systèmes de fichiers journalisés ext3 et ext4.

10.34.1. Installation de E2fsprogs

Modifiez le chemin du répertoire des bibliothèques en lib64 :

```
sed -i '/libdir.*=.*\lib/s@lib@lib64@g' configure
```

La documentation d'E2fsprogs recommande de construire le paquet dans un sous-répertoire du répertoire source :

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

Préparez la compilation d'E2fsprogs :

```
CC="gcc ${BUILD64}" PKG_CONFIG_PATH="${PKG_CONFIG_PATH64}" \
./configure --prefix=/usr --with-root-prefix="" \
--enable-elf-shlibs --disable-libblkid \
--disable-libuuid --disable-fsck \
--disable-uuid
```

Voici la signification des options de configure :

--with-root-prefix=""

Certains programmes (comme **e2fsck** sont considérés 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.

Compilez le paquet :

```
make
```

Pour tester les résultats, exécutez : **make check**.

Installez les binaires, la documentation et les bibliothèques partagées :

```
make install
```

Installez les bibliothèques statiques et les en-têtes :

```
make install-libs
```

10.34.2. 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], libss.[a,so] et libquota.a
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 d'un système de fichiers Linux
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 <code>com_err</code>
debugfs	Un débogueur de système de fichiers. Il est utilisé pour examiner et modifier l'état d'un système de fichiers <code>ext2</code>
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 sur la fragmentation de l'espace libre
e2fsck	Est utilisé pour vérifier, et quelque fois réparer, les systèmes de fichiers <code>ext2</code> , <code>ext3</code> et <code>ext4</code>
e2image	Est utilisé pour sauver les données critiques d'un système de fichiers <code>ext2</code> dans un fichier
e2initrd_helper	Affiche le type de système de fichiers d'un FS donné sur un nom de périphérique ou une étiquette
e2label	Affiche ou modifie le label d'un système de fichiers <code>ext2</code> présent sur un périphérique donné
e2undo	Rejoue un journal annulé pour un système de fichiers <code>ext2/ext3/ext4</code>
e4defrag	Défragmenteur en ligne pour les systèmes de fichiers <code>ext4</code>
filefrag	Renseigne sur le niveau de fragmentation que peut atteindre un fichier
fsck.ext2	Vérifie par défaut les systèmes de fichiers <code>ext2</code> .
fsck.ext3	Vérifie par défaut les systèmes de fichiers <code>ext3</code> .
fsck.ext4	Vérifie par défaut des systèmes de fichiers <code>ext4</code>
fsck.ext4dev	Vérifie par défaut des systèmes de fichiers <code>ext4dev</code>
logsave	Sauvegarde la sortie d'une commande dans un journal applicatif
lsattr	Liste les attributs de fichiers sur un système de fichiers <code>ext2</code> (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 <code>libss</code>
mke2fs	Crée un système de fichiers <code>ext2</code> , <code>ext3</code> ou <code>ext4</code> sur le périphérique donné
mkfs.ext2	Crée par défaut un système de fichiers <code>ext2</code> .
mkfs.ext3	Crée par défaut un système de fichiers <code>ext3</code> .
mkfs.ext4	Crée par défaut un système de fichiers <code>ext4</code> .
mkfs.ext4dev	Crée par défaut un système de fichiers <code>ext4dev</code> .
mklost+found	Est utilisé pour créer un répertoire <code>lost+found</code> sur un système de fichiers <code>ext2</code> ; 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 <code>ext2</code>
tune2fs	Ajuste les paramètres d'un système de fichiers <code>ext2</code>
<code>libcom_err</code>	La routine d'affichage d'erreurs
<code>libe2p</code>	Est utilisé par dumpe2fs , chattr , et lsattr

<code>libext2fs</code>	Contient des routines pour permettre aux programmes du niveau utilisateur de manipuler un système de fichiers <code>ext2</code>
<code>libquota</code>	Fournit une interface pour créer et mettre à jour des fichiers de quota et des champs de superbloc <code>ext4</code>
<code>libss</code>	Est utilisé par debugfs

10.35. Shadow-4.1.5.1

Le paquet Shadow contient des programmes de gestion de mots de passe d'une façon sécurisée.

10.35.1. Installation de Shadow



Remarque

Si vous aimeriez multiplier l'usage des mots de passe efficaces, reportez-vous à <http://cblfs.cross-lfs.org/index.php/Cracklib> 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 ses pages de man, vu que Coreutils a une version meilleure :

```
sed -i 's/groups$(EXEEXT) //' src/Makefile.in
find man -name Makefile.in -exec sed -i '/groups\.1\.xml/d' '{}' \;
find man -name Makefile.in -exec sed -i 's/groups\.1 / /' {} \;
```

Préparez la compilation de Shadow :

```
CC="gcc ${BUILD64}" ./configure --sysconfdir=/etc
```

Voici la signification des options de configure :

`--sysconfdir=/etc`

Dit à Shadow d'installer ses fichiers de configuration dans `/etc` au lieu de `/usr/etc`.

Compilez le paquet :

```
make
```

Ce paquet ne fournit pas de suite de tests.

Installez le paquet :

```
make install
```

Au lieu d'utiliser la méthode *crypt* par défaut, utilisez la méthode *SHA512* 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 /etc/login.defs \
-e 's@#(ENCRYPT_METHOD \).*@\1SHA512@' \
-e 's@/var/spool/mail@/var/mail@'
```



Remarque

Si vous avez construit Shadow avec le support pour Cracklib, exécutez ce **sed** pour corriger le chemin vers le dictionnaire de Cracklib :

```
sed -i 's@DICTPATH.*@DICTPATH\t/lib/cracklib/pw_dict@' /etc/login.defs
```

Déplacez un programme mal placé vers le bon endroit :

```
mv -v /usr/bin/passwd /bin
```

10.35.2. Configurer 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 la commande suivante :

```
grpconv
```

Pour voir ou changer les paramètres par défaut pour les nouveaux comptes utilisateur que vous créez, vous pouvez éditer `/etc/default/useradd`. Voir **man useradd** ou http://cblfs.cross-lfs.org/index.php/Configuring_for_Adding_Users pour plus d'informations.

10.35.3. Configurer le mot de passe de root

Choisissez un mot de passe pour l'utilisateur `root` et configurez-le avec :

```
passwd root
```

10.35.4. Contenu de Shadow

Programmes installés: `chage`, `chfn`, `chpasswd`, `chgrp`, `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` (link to `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
chgrp	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 de séries entière de comptes utilisateur
chsh	Utilisé pour modifier le shell de connexion par défaut d'un utilisateur
expiry	Vérifie et applique 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	Autorise un utilisateur à administrer sa propres liste de membres de son groupe sans avoir besoin des privilèges super-utilisateur

groupmod	Est utilisé pour modifier le nom ou le GID du groupe
grpck	Vérifie l'intégrité des fichiers <code>/etc/group</code> et <code>/etc/gshadow</code>
grpconv	Crée ou met à jour le fichier <code>shadow</code> à partir du fichier <code>group</code> standard
grpunconv	Met à jour <code>/etc/group</code> à partir de <code>/etc/gshadow</code> 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 appliquer 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
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, <code>/etc/passwd</code> et <code>/etc/shadow</code>
pwconv	Crée ou met à jour le fichier de mots de passe <code>shadow</code> à partir du fichier <code>password</code> habituel
pwunconv	Met à jour <code>/etc/passwd</code> à partir de <code>/etc/shadow</code> 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 (<i>User Identification</i> , soit Identification Utilisateur), shell, groupe initial, répertoire personnel et ainsi de suite
vigr	Édite les fichiers <code>/etc/group</code> ou <code>/etc/gshadow</code>
vipw	Édite les fichiers <code>/etc/passwd</code> ou <code>/etc/shadow</code>

10.36. Coreutils-8.20

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

10.36.1. Installation de Coreutils

Un problème connu avec le programme **uname** provenant de ce paquet est que l'option `-p` renvoie toujours `unknown`. Le correctif suivant corrige ce comportement pour toutes les architectures :

```
patch -Np1 -i ../coreutils-8.20-uname-1.patch
```

Maintenant, préparez la compilation de Coreutils :

```
FORCE_UNSAFE_CONFIGURE=1 CC="gcc ${BUILD64}" \  
./configure --prefix=/usr \  
--enable-install-program=hostname
```

Voici la signification des options de configure :

FORCE_UNSAFE_CONFIGURE=1

Oblige Coreutils à se compiler lorsqu'on utilise l'utilisateur `root` user.

Compilez le paquet :

```
make
```

La suite de tests de Coreutils fait plusieurs suppositions sur la présence d'utilisateurs et de groupes système qui ne sont pas valides à l'intérieur de l'environnement minimal existant pour le moment. Donc des étapes supplémentaires doivent être effectuées avant de lancer les tests. Sauter à « Installer le paquet » si vous ne lancez pas la suite de tests.

Créez deux groupes dummy et un utilisateur dummy :

```
echo "dummy1:x:1000:" >> /etc/group  
echo "dummy2:x:1001:dummy" >> /etc/group  
echo "dummy:x:1000:1000::/root:/bin/bash" >> /etc/passwd
```

Tout d'abord, lancez les quelques tests qui ont besoin d'être lancés en tant que `root` :

```
make NON_ROOT_USERNAME=dummy SUBDIRS= check-root
```

Nous allons maintenant lancer la suite de tests en tant qu'utilisateur `dummy`. Corrigez les droits de quelques fichiers pour autoriser cela :

```
chown -Rv dummy .
```

Puis, exécutez le reste des tests en tant qu'utilisateur `nobody` :

```
su dummy -s /bin/bash \  
-c "PATH=$PATH make RUN_EXPENSIVE_TESTS=yes -k check || true"
```

Lorsque les tests sont complétés, supprimez l'utilisateur et les groupes dummy :

```
sed -i '/dummy/d' /etc/passwd /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} /bin
mv -v /usr/bin/{dd,df,echo,false,hostname,ln,ls,mkdir,mknod} /bin
mv -v /usr/bin/{mv,pwd,rm,rmdir,stat,tee,true,uname} /bin
mv -v /usr/bin/chroot /usr/sbin
```

D'autres programmes de Coreutils sont utilisés par certains des scripts du paquet CLFS-Bootscripts. Comme il se peut que /usr ne soit pas disponible pendant les premières étapes du démarrage, ces binaires doivent être sur la partition racine :

```
mv -v /usr/bin/{[,basename,head,install,nice} /bin
mv -v /usr/bin/{readlink,sleep,sync,test,touch} /bin
ln -svf ../../bin/install /usr/bin
```

10.36.2. 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, hostname, 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, and yes

Répertoires installés: libstdbuf.so

Descriptions courtes

base64	base64 encode/décode des données et affiche sur la sortie standard
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	Change le contexte de sécurité pour des fichiers ou des répertoires
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 en échouant. Il renvoie toujours un code d'erreur indiquant l'échec
fmt	Reformate les paragraphes dans les fichiers donnés
fold	Remplit 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)
hostname	Affiche ou initialise le nom de l'hôte
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 vocabulaire 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
mv	Déplace ou renomme des fichiers ou répertoires
nice	Lance un programme avec une priorité modifiée
nl	
nohup	Lance une commande immune aux arrêts brutaux, dont la sortie est redirigée vers le journal de traces

nproc	Affiche le nombre d'unités de calcul disponibles sur le processus actuel
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, à la façon de 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 d'un 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'une échelle et avec un incrément spécifié
sha1sum	Affiche ou vérifie des sommes de contrôle 160-bit Secure Hash Algorithm 1 (SHA1)
sha224sum	Affiche ou vérifie des sommes de contrôle SHA224
sha256sum	Affiche ou vérifie des sommes de contrôle SHA256
sha384sum	Affiche ou vérifie des sommes de contrôle SHA384
sha512sum	Affiche ou vérifie des sommes de contrôle SHA512
shred	Efface les fichiers indiqués en écrivant dessus des modèles aléatoires pour rendre la récupération des données très difficile
shuf	Écrit une permutation aléatoire des lignes en entrée sur la sortie standard ou dans un fichier
sleep	Fait une pause d'un certain temps
sort	Trie les lignes des fichiers donnés
split	Divise les fichiers donnés en plusieurs parties, par taille ou par nombre de lignes
stat	Affiche le statut du fichier ou du système de fichiers
stdbuf	Lance une commande avec des opérations de mise en tampon modifiées pour ses streams standards
stty	Initialise ou affiche les paramètres de la ligne du terminal
sum	Affiche la somme de vérification et le nombre de blocs pour chacun des fichiers de données
sync	Vide les tampons du système de fichiers. Cela force l'enregistrement des blocs sur disque 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 des fichiers indiqués
test ou [Compare les valeurs et vérifie les types de fichiers

timeout	Lance une commande avec une limite temporelle
touch	Modifie les dates et heures du 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	Traduit, réduit et supprime les caractères donnés à partir de l'entrée standard
true	Ne fait rien mais avec succès. Il quitte avec un code de sortie indiquant une réussite
truncate	Réduit ou agrandit un fichier à 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 les informations système
unexpand	Convertit les espaces en tabulations
uniq	Conserve une ligne parmi plusieurs lignes identiques successives
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 « y » ou la chaîne précisée de manière répétée jusqu'à être tué
libstdbuf	Bibliothèque utilisée par stdbuf

10.37. Iana-Etc-2.30

Le paquet Iana-Etc fournit des données pour les services et protocoles réseau.

10.37.1. Installation de Iana-Etc



Remarque

Ce paquet comporte une option pour télécharger les données mises à jour quand un accès internet est disponible. Si `/etc/resolv.conf` a une entrée `nameserver` et si un accès internet est disponible à ce moment, appliquez le correctif IANA get et mettez à jour les données :

```
patch -Np1 -i ../iana-etc-2.30-get_fix-1.patch
```

```
make get
```

N'appliquez pas le correctif suivant.

Le correctif suivant met à jour les fichiers services et de protocole :

```
patch -Np1 -i ../iana-etc-2.30-numbers_update-20120610-2.patch
```

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 ne fournit pas de suite de tests.

Installez le paquet :

```
make install
```

10.37.2. Contenu de Iana-Etc

Fichier installés: `/etc/protocols` and `/etc/services`

Descriptions courtes

<code>/etc/protocols</code>	Décrit les différents protocoles Internet DARPA disponibles à partir du sous-système TCP/IP
<code>/etc/services</code>	Fournit une correspondance entre des noms de services internet et leur numéros de port et types de protocoles affectés

10.38. M4-1.4.16

Le paquet M4 contient un processeur de macros.

10.38.1. Installation de M4

Préparez la compilation de M4 :

```
CC="gcc ${BUILD64}" ./configure --prefix=/usr
```

Compilez le paquet :

```
make
```

Pour tester les résultats, exécutez : **make check**.

Installez le paquet :

```
make install
```

10.38.2. 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.

10.39. Bibliothèques Bison-2.6.4 32 bits

Le paquet Bison contient un générateur d'analyseurs.

10.39.1. Installation de Bison

Le script **configure** ne détermine pas la bonne valeur pour la suite. Définissez la valeur à la main :

```
echo "ac_cv_prog_lex_is_flex=yes" > config.cache
```

Préparez la compilation de Bison :

```
CC="gcc ${BUILD32}" ./configure --prefix=/usr --cache-file=config.cache
```

Compilez le paquet :

```
make
```

Pour tester les résultats, exécutez : **make check**.

Installez le paquet :

```
make install
```

Les détails sur ce paquet sont disponibles dans Section 10.40.2, « Contenu de Bison. »

10.40. Bison-2.6.4 64 bits

Le paquet Bison contient un générateur d'analyseurs.

10.40.1. Installation de Bison

Le script **configure** ne détermine pas la bonne valeur pour la suite. Définissez la valeur à la main :

```
echo "ac_cv_prog_lex_is_flex=yes" > config.cache
```

Préparez la compilation de Bison :

```
CC="gcc ${BUILD64}" ./configure --prefix=/usr --libdir=/usr/lib64 --cache-file=
```

Compilez le paquet :

```
make
```

Pour tester les résultats, exécutez : **make check**.

Installez le paquet :

```
make install
```

10.40.2. Contenu de Bison

Programmes installés:	bison and yacc
Répertoire installé:	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 remplaçant de Yacc (Yet Another Compiler Compiler)
yacc	Une enveloppe 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 <i>yyerror</i> et <i>main</i> . Cette bibliothèque n'est généralement pas très utile mais POSIX la réclame

10.41. Libtool-2.4.2 32 bit Libraries

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.

10.41.1. Installation de Libtool

Le fichier `config.cache` suivant permet de forcer le chemin de recherche par défaut pour prendre en compte l'environnement multilib :

```
echo "lt_cv_sys_dlsearch_path='/lib /usr/lib /usr/local/lib /opt/lib'" > config.cache
```

Préparez la compilation de Libtool :

```
CC="gcc ${BUILD32}" ./configure --prefix=/usr \
  --cache-file=config.cache
```

Compilez le paquet :

```
make
```

Pour tester les résultats, exécutez : **make LDEMULATION=elf_i386 check.**

Voici la signification du remplacement de **make check** :

```
LDEMULATION=elf_i386
```

Libtool a tendance à se comporter de façon inattendue en environnement multilib, du moins sur des architectures autres que celle par défaut. On ne saisit pas tout à fait les causes de ces erreurs car elles peuvent apparaître ou disparaître bon gré mal gré, même lors de changements normalement inoffensifs au bon déroulement de la compilation. Dans cette version du livre, l'un des tests, `pdemo-make`, ne se lie pas correctement car il cherche à lier les objets 32 bits aux bibliothèques 64 bits du système. Cette option permet à ce test de réussir sans impacter les autres tests (par rapport aux méthodes de correction plus traditionnelles comme spécifier `LD="gcc ${BUILD32}"`, ce qui empêche l'exécution de nombreux test ou bien configurer avec `LDFLAGS='-L/lib -L/usr/lib'`, ce qui dans ce cas fait échouer d'autres tests).

Installez le paquet :

```
make install
```

Préparez `libtool` à être pris en charge par le programme enveloppe multi-architectures (« multiarch-wrapper »). Libtool seul ne supporte pas l'environnement multilib :

```
mv -v /usr/bin/libtool{,-32}
```

Les détails sur ce paquet sont disponibles dans Section 10.42.2, « Contenu de Libtool. »

10.42. Libtool-2.4.2 64 bits

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.

10.42.1. Installation de Libtool

Le fichier `config.cache` suivant permet de forcer le chemin de recherche par défaut pour prendre en compte l'environnement multilib :

```
echo "lt_cv_sys_dlsearch_path='/lib64 /usr/lib64 /usr/local/lib64 /opt/lib64'"
```

Préparez la compilation de Libtool :

```
CC="gcc ${BUILD64}" ./configure --prefix=/usr \
  --libdir=/usr/lib64 --cache-file=config.cache
```

Compilez le paquet :

```
make
```

Pour tester les résultats, exécutez : **make check**.

Installez le paquet :

```
make install
```

Préparez `libtool` à être pris en charge par le programme enveloppe multi-architectures (« multiarch-wrapper »). Libtool seul ne supporte pas l'environnement multilib :

```
mv -v /usr/bin/libtool{,-64}
ln -sv multiarch_wrapper /usr/bin/libtool
```

10.42.2. Contenu de Libtool

Programmes installés:	libtool and 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

10.43. Bibliothèques Flex-2.5.37 32 bits

Le paquet Flex contient un outil de génération de programmes reconnaissant des motifs de texte.

10.43.1. Installation de Flex

Préparez la compilation de Flex :

```
CC="gcc ${BUILD32}" ./configure --prefix=/usr
```

Compilez le paquet :

```
make libfl.a
```

Installez le paquet :

```
make install-libLIBRARIES
```

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
```

Les détails sur ce paquet sont disponibles dans Section 10.44.2, « Contenu de Flex. »

10.44. Flex-2.5.37 64 bits

Le paquet Flex contient un outil de génération de programmes reconnaissant des motifs de texte.

10.44.1. Installation de Flex

Préparez la compilation de Flex :

```
CC="gcc ${BUILD64}" ./configure --prefix=/usr \
  --libdir=/usr/lib64
```

Compilez le paquet :

```
make
```

Pour tester les résultats, exécutez : **make check**.

Installez le paquet :

```
make install
```

Certains paquets s'attendent à trouver la bibliothèque `lex` dans `/usr/lib64`. Pour cela, créez un lien symbolique :

```
ln -sv libfl.a /usr/lib64/libl.a
```

Quelques programmes ne connaissent pas encore **flex** et essaient de lancer son prédécesseur, **lex**. Pour ces programmes, créez un script enveloppe nommé `lex` appelant `flex` en mode d'émulation **lex** :

```
cat > /usr/bin/lex << "EOF"
#!/bin/sh
# Début de /usr/bin/lex

exec /usr/bin/flex -l "$@"

# Fin de /usr/bin/lex
EOF
chmod -v 755 /usr/bin/lex
```

10.44.2. Contenu de Flex

Programmes installés: flex and lex
Bibliothèques installées: libfl.a et libfl_pic.a

Descriptions courtes

flex	Un outil pour générer des programmes reconnaissant des motifs dans un texte. Il permet une grande flexibilité pour spécifier les règles de recherche de motif, supprimant ainsi le besoin de développer un programme spécialisé
flex++	Lien vers flex qui lui fait générer du des classes d'analyse C++
lex	Un script qui exécute flex en mode d'émulation lex
libfl.a	La bibliothèque flex
libfl_pic.a	La bibliothèque flex

10.45. IPRoute2-3.4.0

Le paquet IPRoute2 contient des programmes pour le réseau, basique ou avancé, basé sur IPV4.

10.45.1. Installation de IPRoute2

Par défaut, ce paquet construit le programme **arpd** qui dépend de Berkeley DB. Vu que **arpd** n'est pas une exigence vraiment courante sur un système Linux, supprimez la dépendance de Berkeley DB en utilisant les commandes ci-dessous. Si vous avez besoin du binaire **arpd**, vous pouvez trouver des instructions pour compiler Berkeley DB dans le livre CBLFS sur http://cblfs.cross-lfs.org/index.php/Berkeley_DB.

```
sed -i '/^TARGETS/s@arpd@g' misc/Makefile
sed -i '/ARPD/d' Makefile
rm -v man/man8/arpd.8
```

Supprimez les en-têtes libnl inutilisés :

```
sed -i '/netlink\\//d' ip/ipl2tp.c
```

Le correctif suivant permet de mettre à jour le contenu de LIBDIR :

```
patch -Np1 -i ../iproute2-3.4.0-libdir-1.patch
```

Compilez le paquet :

```
make CC="gcc ${BUILD64}" DESTDIR= LIBDIR=/usr/lib64 \
    DOCDIR=/usr/share/doc/iproute2 MANDIR=/usr/share/man
```

Voici la signification de l'option de make :

DESTDIR=

Cette option remplace le DESTDIR de /usr par défaut afin que les binaires IPRoute2 soient installés dans /sbin. C'est le bon emplacement suivant la FHS parce que certains des binaires IPRoute2 sont utilisés dans le paquet LFS-Bootscripts.

DOCDIR=/usr/share/doc/iproute2 MANDIR=/usr/share/man

Il peut résulter du paramètre DESTDIR=/ que la documentation soit installée dans /share/doc et dans /share/man. Ces options assurent que les docs sont installées aux bons endroits.

Ce paquet ne fournit pas de suite de tests.

Installez le paquet :

```
make DESTDIR= LIBDIR=/usr/lib64 \
    DOCDIR=/usr/share/doc/iproute2 \
    MANDIR=/usr/share/man install
```

10.45.2. Contenu de IPRoute2

Programmes installés:	ctstat (link to Instat), genl, ifcfg, ifstat, ip, Instat, nstat, routef, routel, rtacct, rtmon, rtpr, rtstat (link to Instat), ss, et tc
Répertoires installés:	/etc/iproute2, /lib/tc, /usr/lib/tc, /usr/share/doc/iproute2

Descriptions courtes

ctstat	Outil donnant le statut de la connexion
genl	Needs description

ifcfg	Un emballage en script shell pour la commande ip .
ifstat	Affiche les statistiques des interfaces, incluant le nombre de paquets émis et transmis par l'interface
ip	<p>L'exécutable principal. Il a plusieurs fonctions :</p> <p>ip link [périphérique] permet aux utilisateurs de regarder l'état des périphériques et de faire des changements.</p> <p>ip addr permet aux utilisateurs de regarder les adresses et leurs propriétés, d'ajouter de nouvelles adresse et de supprimer les anciennes.</p> <p>ip neighbor permet aux utilisateurs de regarder dans les liens des voisins et dans les leurs, d'ajouter de nouvelles entrées et de supprimer les anciennes.</p> <p>ip rule permet aux utilisateurs de regarder les politiques de routage et de les modifier.</p> <p>ip route permet aux utilisateurs de regarder la table de routage et de modifier les règles de routage.</p> <p>ip tunnel permet aux utilisateurs de regarder les tunnels IP et leurs propriétés, et de les modifier.</p> <p>ip maddr permet aux utilisateurs de regarder les adresses multicast et leurs propriétés, et de les changer.</p> <p>ip mroute permet aux utilisateurs de configurer, modifier ou supprimer le routage multicast.</p> <p>ip monitor permet aux utilisateurs de surveiller en permanence l'état des périphériques, des adresses et des routes.</p>
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 <code>/proc/net/route</code>
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	<p>Exécutable de contrôle du trafic ; utile pour l'implémentation de la qualité de service (QOS) et de la classe de service (COS)</p> <p>tc qdisc permet aux utilisateurs de configurer la discipline de queues</p> <p>tc class permet aux utilisateurs de configurer les classes suivant la planification de la discipline de queues</p> <p>tc estimator autorise les utilisateurs d'estimer le flux réseau dans un réseau</p> <p>tc filter permet aux utilisateurs de configurer les filtres de paquets pour QOS/COS</p> <p>tc policy permet aux utilisateurs de configurer les politiques QOS/COS</p>

10.46. Bibliothèques Perl-5.16.2 32 bits

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

10.46.1. Installation de Perl

Par défaut, le module `Compress::Raw::Zlib` de Perl se construit et se lie à sa propre copie de `zlib`. La commande suivante lui dit d'utiliser la `zlib` installée sur le système :

```
sed -i -e '/^BUILD_ZLIB/s/True/False/' \
-e '/^INCLUDE/s,\./zlib-src,/usr/include,' \
-e '/^LIB/s,\./zlib-src,/usr/lib,' \
cpan/Compress-Raw-Zlib/config.in
```



Remarque

Si vous suivez la méthode du démarrage, vous aurez besoin d'activer le périphérique loopback et de paramétrer le nom de l'hôte (*hostname*) pour certains des tests :

```
ip link set lo up
hostname clfs
```

Avant de lancer la configuration, créez un fichier `/etc/hosts` basique qui va être d'une part référencé par un des fichiers de configuration de Perl et d'autre part utilisé par la suite de tests :

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

Pour avoir un contrôle complet de la façon dont Perl est paramétré, vous pouvez lancer le script **Configure** et choisir la façon dont ce paquet est construit. Si vous préférez plutôt utiliser les paramètres par défaut autodétectés par Perl, préparez la compilation de Perl avec :

```
./configure.gnu --prefix=/usr \
-Dvendorprefix=/usr \
-Dman1dir=/usr/share/man/man1 \
-Dman3dir=/usr/share/man/man3 \
-Dpager="/bin/less -isR" \
-Dcc="gcc ${BUILD32}" \
-Dsethreads -Duseshrplib
```

Voici la signification de l'option de configure :

`-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.gnu** pense que nous ne voulons pas les pages de man de Perl. Ces paramètres changent cette décision.

`-Dsethreads`

Ceci dit à Perl d'utiliser les threads.

`-Duseshrplib`

Ceci dit à Perl de construire une `libperl` partagée.

Compilez le paquet :

```
make
```

Pour tester les résultats, exécutez : **make test**.

Installez le paquet :

```
make install
```

Ajoutez un suffixe au binaire **perl** afin de l'emballer avec le multiarch wrapper :

```
mv -v /usr/bin/perl{,-32}  
mv -v /usr/bin/perl5.16.2{,-32}
```

Les détails sur ce paquet sont disponibles dans Section 10.47.2, « Contenu de Perl. »

10.47. Perl-5.16.2 64 bits

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

10.47.1. Installation de Perl

Par défaut, le module `Compress::Raw::Zlib` de Perl se construit et se lie à sa propre copie de de Zlib. La commande suivante lui dit d'utiliser la Zlib installée sur le système :

```
sed -i -e '/^BUILD_ZLIB/s/True/False/' \
-e '/^INCLUDE/s,\./zlib-src,/usr/include,' \
-e '/^LIB/s,\./zlib-src,/usr/lib64,' \
cpan/Compress-Raw-Zlib/config.in
```

Perl n'a pas connaissance par défaut de noms des répertoires de bibliothèques autres que `lib`. Le correctif suivant va permettre son installation dans d'autres répertoires :

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

Une autre anomalie subsiste, bien que moins importante : si nous installons perl et lançons la commande `perl -V`, elle affichera que la `libc` est dans `/lib`. La commande `sed` suivante corrige ce comportement mais ne prendra effet que lors de la commande `make install` :

```
sed -i "/libc/s@/lib@/lib64@" hints/linux.sh
```

Nous devons encore dire à Perl d'utiliser le répertoire `lib64` :

```
echo 'installstyle="lib64/perl5"' >>hints/linux.sh
```

Pour avoir un contrôle complet de la façon dont Perl est paramétré, vous pouvez lancer le script **Configure** et choisir la façon dont ce paquet est construit. Si vous préférez plutôt utiliser les paramètres par défaut autodétectés par Perl, préparez la compilation de Perl avec :

```
./configure.gnu --prefix=/usr \
-Dvendorprefix=/usr \
-Dman1dir=/usr/share/man/man1 \
-Dman3dir=/usr/share/man/man3 \
-Dpager="/bin/less -isR" \
-Dlibpth="/usr/local/lib64 /lib64 /usr/lib64" \
-Dcc="gcc ${BUILD64}" \
-Dusetthreads -Duseshrplib
```

Voici la signification des options de configure :

`-Dlibpth="/usr/local/lib64 /lib64 /usr/lib64"`

Ceci indique à Perl de se lier aux bibliothèques 64 bits.

`-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.gnu** pense que nous ne voulons pas les pages de man de Perl. Ces paramètres changent cette décision.

`-Dusetthreads`

Ceci dit à Perl d'utiliser les threads.

`-Duseshrplib`

Ceci dit à Perl de construire une libperl partagée.

Compilez le paquet :

```
make
```

Pour tester les résultats, exécutez : **make test**.

Installez le paquet :

```
make install
```

Ajoutez un suffixe au binaire **perl** afin de l'emballer avec le multiarch wrapper :

```
mv -v /usr/bin/perl{,-64}
mv -v /usr/bin/perl5.16.2{,-64}
```

Nous devons à présent créer un lien vers le multiarch wrapper afin de nous permettre de choisir quelle installation de Perl utiliser :

```
ln -sv multiarch_wrapper /usr/bin/perl
ln -sv multiarch_wrapper /usr/bin/perl5.16.2
```

La valeur de la variable d'environnement `USE_ARCH` décidera du binaire de perl à exécuter. **USE_ARCH=32 perl -V:cc** donnera la valeur du CC utilisé pour construire Perl 32 bits. Le `multiarch_wrapper` sera utile plus tard pour compiler des extensions de Perl. Sans le programme `multiarch_wrapper`, il sera très difficile de mettre en place une extension 32 bits.

10.47.2. 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, perlvp, 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, and zipdetails

Bibliothèques installées: Plusieurs centaines qu'on ne peut pas tous lister ici.

Répertoires installés: /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	Remplace ou modifie la configuration de modules Perl
corelist	Une interface en ligne de commande pour <code>Module::CoreList</code>
cpan	Script shell qui fournit une interface de commande à CPAN.pm.
cpan2dist	Le créateur de distribution CPANPLUS
cpanp	Le lanceur CPANPLUS
cpanp-run-perl	Script Perl qui (description needed)
enc2xs	Construit une extension Perl pour le module Encode, soit à partir de <i>Unicode Character Mappings</i> soit à partir de <i>Tcl Encoding Files</i>
find2perl	Traduit les commandes find en Perl

h2ph	Convertit les fichiers d'en-têtes C <code>.h</code> en fichiers d'en-têtes Perl <code>.ph</code>
h2xs	Convertit les fichiers d'en-têtes C <code>.h</code> en extensions Perl
instmodsh	Un script shell pour observer les modules Perl installés, il peut 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 <code>libnet</code>
ptar	Un programme similaire à tar écrit en Perl
ptardiff	Un programme Perl qui compare une archive extraite et une autre non extraite
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
perlvp	La procédure de vérification d'installation de Perl (<i>Perl Installation Verification Procedure</i>). Elle peut être utilisée pour vérifier que Perl et ses bibliothèques ont été installés correctement
perlthanks	Utilisé pour générer des messages de remerciement aux développeurs de Perl
piconv	Une version Perl du convertisseur d'encodage des caractères iconv
pl2pm	Un outil simple pour la conversion des fichiers Perl4 <code>.pl</code> en modules Perl5 <code>.pm</code>
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 <code>*roff</code>
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	Un outil en ligne de commande pour lancer des tests liés au module <code>Test::Harness</code> .
psed	Une version Perl de l'éditeur de flux sed
pstruct	Affiche les structures C générées à partir de cc -g -S stabs
ptargrep	Un programme Perl appliquant des modèles correspondant au contenu des fichiers d'une archive tar
s2p	Traduit 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

10.48. Bibliothèques Readline-6.2 32 bits

Le paquet Readline est un ensemble de bibliothèques qui offrent des fonctionnalités d'édition de la ligne de commande et d'historique.

10.48.1. Installation de Readline

Le correctif suivant contient des mises à jour issues du mainteneur. Le mainteneur de Readline ne fait ces correctifs que pour corriger des problèmes sérieux :

```
patch -Np1 -i ../readline-6.2-branch_update-3.patch
```

Préparez la compilation de Readline:

```
CC="gcc ${BUILD32}" CXX="g++ ${BUILD32}" \  
./configure --prefix=/usr --libdir=/lib
```

Compilez le paquet :

```
make SHLIB_LIBS=-lncurses
```

Ce paquet ne fournit pas de suite de tests.

Installez le paquet :

```
make install
```

Installez la documentation :

```
make install-doc
```

Maintenant, déplacez les bibliothèques statiques vers un endroit plus approprié :

```
mv -v /lib/lib{readline,history}.a /usr/lib
```

Ensuite, supprimez les fichiers the .so dans /lib et liez-les à nouveau dans /usr/lib.

```
rm -v /lib/lib{readline,history}.so  
ln -svf ../../lib/libreadline.so.6 /usr/lib/libreadline.so  
ln -svf ../../lib/libhistory.so.6 /usr/lib/libhistory.so
```

Les détails sur ce paquet sont disponibles dans Section 10.49.2, « Contenu de Readline. »

10.49. Readline-6.2 64 bits

Le paquet Readline est un ensemble de bibliothèques qui offrent des fonctionnalités d'édition de la ligne de commande et d'historique.

10.49.1. Installation de Readline

Le correctif suivant contient des mises à jour issues du mainteneur. Le mainteneur de Readline ne fait ces correctifs que pour corriger des problèmes sérieux :

```
patch -Np1 -i ../readline-6.2-branch_update-3.patch
```

Préparez la compilation de Readline:

```
CC="gcc ${BUILD64}" CXX="g++ ${BUILD64}" \
./configure --prefix=/usr --libdir=/lib64
```

Compilez le paquet :

```
make SHLIB_LIBS=-lcurses
```

Ce paquet ne fournit pas de suite de tests.

Installez le paquet :

```
make install
```

Installez la documentation :

```
make install-doc
```

Maintenant, déplacez les bibliothèques statiques vers un endroit plus approprié :

```
mv -v /lib64/lib{readline,history}.a /usr/lib64
```

Ensuite, supprimez les fichiers .so dans /lib64 et recréez les liens dans /usr/lib64.

```
rm -v /lib64/lib{readline,history}.so
ln -svf ../../lib64/libreadline.so.6 /usr/lib64/libreadline.so
ln -svf ../../lib64/libhistory.so.6 /usr/lib64/libhistory.so
```

10.49.2. Contenu de Readline

Bibliothèques installées: libhistory.[a,so] et libreadline.[a,so]
Répertoires installés: /usr/include/readline, /usr/share/readline

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 de petits programmes qui ont besoin d'une interface en ligne de commande

10.50. Autoconf-2.69

Le paquet Autoconf contient des programmes produisant des scripts shell qui configurent automatiquement le code source.

10.50.1. Installation de Autoconf

Préparez la compilation d'Autoconf :

```
CC="gcc ${BUILD64}" ./configure --prefix=/usr
```

Compilez le paquet :

```
make
```

Pour tester les résultats, exécutez : **make check VERBOSE=yes**. 17 tests sont sautés, ils utilisent Automake et des langages différents de GCC. Pour l'accomplissement des tests, vous pouvez retester Autoconf après qu'Automake a été installé.

Installez le paquet :

```
make install
```

10.50.2. 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 le code source des paquets, 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 <i>#define</i> que configure utilise.
autom4te	Une enveloppe 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 <code>configure.in</code> pour un paquet logiciel. Il examine les fichiers source d'un répertoire et crée un fichier <code>configure.scan</code> servant de fichier <code>configure.in</code> préliminaire pour le paquet
autoupdate	Modifie un fichier <code>configure.in</code> 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 <code>configure.in</code> pour un paquet logiciel. Il affiche les identifiants que le paquet utilise dans des conditions du préprocesseur C. Si un paquet a déjà été initialisé pour avoir une certaine portabilité, ce programme aide à déterminer ce que configure doit vérifier. Il peut aussi remplir les blancs dans un fichier <code>configure.in</code> généré par autoscan

10.51. Automake-1.12.4

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

10.51.1. Installation de Automake

Préparez la compilation d'Automake :

```
CC="gcc ${BUILD64}" ./configure --prefix=/usr
```

Compilez le paquet :

```
make
```

Pour tester les résultats, exécutez : **make check**.

Installez le paquet :

```
make install
```

10.51.2. Contenu de Automake

Programmes installés: acinstall, aclocal, aclocal-1.12, automake, automake-1.12, 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-1.12, /usr/share/automake-1.12, /usr/share/doc/automake

Descriptions courtes

acinstall	Un script qui installe des fichiers M4, style aclocal
aclocal	Génère des fichiers <code>aclocal.m4</code> basés sur le contenu du fichier <code>configure.in</code>
aclocal-1.12.4	Un lien vers aclocal
automake	Un outil pour générer automatiquement des fichiers <code>Makefile.in</code> à partir de fichiers <code>Makefile.am</code> . Pour créer tous les fichiers <code>Makefile.in</code> d'un paquet, lancez ce programme dans le répertoire de haut niveau. En parcourant le fichier <code>configure.in</code> , il trouve automatiquement chaque fichier <code>Makefile.am</code> approprié et génère le fichier <code>Makefile.in</code>
automake-1.12	Un lien en dur vers automake
compile	Une enveloppe pour les compilateurs
config.guess	Un script qui tente de deviner le 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

Une enveloppe pour **lex** et **yacc**

10.52. Bash-4.2

Le paquet Bash contient le shell Bourne-Again.

10.52.1. Installation de Bash

Le correctif suivant contient des mises à jour issues du mainteneur. Le mainteneur de Bash ne fait ces correctifs que pour corriger des problèmes sérieux :

```
patch -Np1 -i ../bash-4.2-branch_update-6.patch
```

La commande sed suivante modifie configure pour qu'il cherche le bibliothèque Readline dans le bon répertoire :

```
sed -i "/ac_cv_rl_libdir/s@/lib@&64@" configure
```

Préparez la compilation de Bash :

```
CC="gcc ${BUILD64}" CXX="g++ ${BUILD64}" \
./configure --prefix=/usr --bindir=/bin \
--without-bash-malloc --with-installed-readline
```

Voici la signification de l'options de configure :

--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
```

Pour tester les résultats, exécutez : **make tests**.

Installez le paquet :

```
make htmdir=/usr/share/doc/bash-4.2 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 programmes soient découverts au fur et à mesure de leur disponibilité.

10.52.2. Contenu de Bash

Programmes installés: bash, bashbug et sh (link to 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 au standard POSIX

10.53. Bibliothèques Bzip2-1.0.6 32 bits

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 le classique **gzip**.

10.53.1. Installation de Bzip2

Par défaut bzip2 crée des liens symboliques qui utilisent des noms de chemins absolus. Le sed suivant fera en sorte que qu'ils soient créés plutôt avec des chemins relatifs :

```
sed -i -e 's:ln -s -f $(PREFIX)/bin/:ln -s :' Makefile
```

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

```
make -f Makefile-libbz2_so CC="gcc ${BUILD32}" CXX="g++ ${BUILD32}"
make clean
```

L'option `-f` va faire que Bzip2 sera compilé en utilisant un fichier `Makefile`, dans ce cas le fichier `Makefile-libbz2_so`, qui crée une bibliothèque dynamique `libbz2.so` et lie les outils de Bzip2 contre elle.

Recompilez le paquet sans utiliser la bibliothèque partagée :

```
make CC="gcc ${BUILD32}" CXX="g++ ${BUILD32}" libbz2.a
```

Pour tester les résultats, lancez : `make CC="gcc ${BUILD32}" CXX="g++ ${BUILD32}" check`.

Installez les bibliothèques et créez un lien symbolique nécessaire :

```
cp -v libbz2.a /usr/lib
cp -av libbz2.so* /lib
ln -sv ../../lib/libbz2.so.1.0 /usr/lib/libbz2.so
```

Les détails sur ce paquet sont disponibles dans Section 10.54.2, « Contenu de Bzip2. »

10.54. Bzip2-1.0.6 64 bits

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 le classique **gzip**.

10.54.1. Installation de Bzip2

Par défaut bzip2 crée des liens symboliques qui utilisent des noms de chemins absolus. Le sed suivant fera en sorte que qu'ils soient créés plutôt avec des chemins relatifs :

```
sed -i -e 's:ln -s -f $(PREFIX)/bin/:ln -s : ' Makefile
```

Il nous faut modifier le chemin par défaut des bibliothèques en `lib64` :

```
sed -i 's@/lib\(/|\ | \$\)@/lib64\1@g' Makefile
```

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

```
make -f Makefile-libbz2_so CC="gcc ${BUILD64}" CXX="g++ ${BUILD64}"
make clean
```

L'option `-f` va faire que Bzip2 sera compilé en utilisant un fichier `Makefile`, dans ce cas le fichier `Makefile-libbz2_so`, qui crée une bibliothèque dynamique `libbz2.so` et lie les outils de Bzip2 contre elle.

Recompilez le paquet en utilisant une bibliothèque non partagée et testez-le :

```
make CC="gcc ${BUILD64}" CXX="g++ ${BUILD64}"
```

Installez les programmes :

```
make CC="gcc ${BUILD64}" CXX="g++ ${BUILD64}" PREFIX=/usr install
```

Installez le binaire partagé **bzip2** dans le répertoire `/bin`, faites quelques liens symboliques nécessaires et nettoyez :

```
cp -v bzip2-shared /bin/bzip2
cp -av libbz2.so* /lib64
ln -sv ../../lib64/libbz2.so.1.0 /usr/lib64/libbz2.so
rm -v /usr/bin/{bunzip2,bzcat,bzip2}
ln -sv bzip2 /bin/bunzip2
ln -sv bzip2 /bin/bzcat
```

10.54.2. 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 bzmores) et bzmores
Bibliothèques installées:	libbz2.a, libbz2.so (link to libbz2.so.1.0), libbz2.so.1.0 (link to libbz2.so.1.0.6) et libbz2.so.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	Comprime les fichiers en utilisant l'algorithme de compression de texte par tri de blocs de Burrows-Wheeler avec le codage de 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

10.55. Diffutils-3.2

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

10.55.1. Installation de Diffutils

Préparez la compilation de Diffutils :

```
CC="gcc ${BUILD64}" ./configure --prefix=/usr
```

Diffutils veut **ed** comme éditeur. Le sed suivant va nous permettre d'utiliser vim :

```
sed -i 's@\(^\#define DEFAULT_EDITOR_PROGRAM \).*@1"vi"@' lib/config.h
```

Compilez le paquet :

```
make
```

Pour tester les résultats, lancez : **make check**.

Installez le paquet :

```
make install
```

10.55.2. 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

10.56. Bibliothèques File-5.11 32 bits

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

10.56.1. Installation de File

Préparez la compilation de File :

```
CC="gcc ${BUILD32}" ./configure --prefix=/usr
```

Compilez le paquet :

```
make
```

Ce paquet ne fournit pas de suite de tests.

Installez le paquet :

```
make install
```

Les détails sur ce paquet sont disponibles dans Section 10.57.2, « Contenu de File. »

10.57. File-5.11 64 bits

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

10.57.1. Installation de File

Préparez la compilation de File :

```
CC="gcc ${BUILD64}" ./configure --prefix=/usr \
  --libdir=/usr/lib64
```

Compilez le paquet :

```
make
```

Ce paquet ne fournit pas de suite de tests.

Installez le paquet :

```
make install
```

10.57.2. Contenu de File

Programmes installés:	file
Répertoire installé:	libmagic.[a,so]

Descriptions courtes

file	Tente de classer 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

10.58. Gawk-4.0.1

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

10.58.1. Installation de Gawk

Préparez la compilation de Gawk :

```
CC="gcc ${BUILD64}" ./configure --prefix=/usr \
  --libexecdir=/usr/lib64
```

Compilez le paquet :

```
make
```

Pour tester les résultats, exécutez : **make check**.

Installez le paquet :

```
make install
```

10.58.2. Contenu de Gawk

Programmes installés:	awk (link to gawk), gawk, gawk-4.0.1, grcat, igawk, pgawk, pgawk-4.0.1 et pwcat
Répertoire installé:	/usr/lib/awk, /usr/share/awk

Descriptions courtes

awk	Un lien vers gawk
gawk	Un programme de manipulation de fichiers texte. C'est l'implémentation GNU de awk
gawk-4.0.1	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-4.0.1	Lien en dur vers pgawk
pwcat	Affiche la base de données de mots de passe /etc/passwd

10.59. 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).

10.59.1. Installation de Findutils

Préparez la compilation de Findutils :

```
CC="gcc ${BUILD64}" ./configure --prefix=/usr \
  --libexecdir=/usr/lib64/locate --localstatedir=/var/lib64/locate
```

Voici la signification des options de configure :

`--localstatedir`

Cette option modifie l'emplacement de la base de données **locate** dans `/var/lib64/locate`, qui est conforme au FHS.

Compilez le paquet :

```
make
```

Pour tester les résultats, exécutez : **make check**.

Installez le paquet :

```
make install
```

Le programme **find** est utilisé par certains des scripts du paquet CLFS-Bootscripts. Comme il se peut que `/usr` ne soit pas disponible pendant les premières étapes du démarrage, le binaire **find** doit être sur la partition racine :

```
mv -v /usr/bin/find /bin
```

Le script **updatedb** doit être modifié pour pointer vers le nouvel emplacement de **find** :

```
sed -i 's@find:=${BINDIR}@find:="/bin@' /usr/bin/updatedb
```

10.59.2. Contenu de Findutils

Programmes installés: bigram, code, find, frcode, locate, updatedb, and xargs

Descriptions courtes

bigram	Anciennement utilisé pour générer les bases de données locate
code	Anciennement utilisé pour générer les bases de données locate . C'est l'ancêtre de frcode .
find	Recherche dans des répertoires donnés des fichiers selon certains critères
frcode	est appelé par updatedb pour compresser la liste des noms de fichiers. La compression se fait à la volée, divisant la taille de la base par quatre ou cinq.
locate	Recherche dans une base de données de noms de fichiers une chaîne un motif donné
updatedb	Met à jour la base de données locate . Elle scanne le système de fichiers entier (y compris les autres systèmes de fichiers montés, sauf si on lui spécifie de ne pas les prendre en compte) puis enregistre chaque nom de fichier trouvé dans la base de données
xargs	Peut être utilisé pour appliquer une commande à une liste de fichiers

10.60. Bibliothèques Gettext-0.18.1.1 32 bits

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.

10.60.1. Installation de Gettext

Préparez la compilation de Gettext :

```
CC="gcc ${BUILD32}" CXX="g++ ${BUILD32}" \  
./configure --prefix=/usr
```

Compilez le paquet :

```
make
```

Pour tester les résultats, exécutez : **make check**.

Installez le paquet :

```
make install
```

Les détails sur ce paquet sont disponibles dans Section 10.61.2, « Contenu de Gettext. »

10.61. Gettext-0.18.1.1 64 bits

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.

10.61.1. Installation de Gettext

Préparez la compilation de Gettext :

```
CC="gcc ${BUILD64}" CXX="g++ ${BUILD64}" \
./configure --prefix=/usr --libdir=/usr/lib64
```

Compilez le paquet :

```
make
```

Pour tester les résultats, exécutez : **make check**.

Installez le paquet :

```
make install
```

10.61.2. 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, and preloadable_libintl.so
Répertoires installés:	/usr/lib/gettext, /usr/share/doc/gettext, /usr/share/gettext

Descriptions courtes

autopoint	Copie les fichiers d'infrastructure standard gettext en un paquet source
config.charset	Sort un tableau dépendant du système des alias d'encodage.
config.rpath	Sort un ensemble de variables dépendant du système décrivant comment régler le chemin de recherche au moment de l'exécution des bibliothèques partagées dans un exécutable.
envsubst	Substitue les variables d'environnement dans des chaînes formatées 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 dans 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 <code>.po</code>
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 <code>.po</code> , initialise l'environnement de l'utilisateur
msgmerge	Fusionne 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'écriture cyrillique au latin
xgettext	Extrait les lignes de messages traduisibles à partir des fichiers source donnés pour réaliser la première traduction de modèle
<code>libasprintf</code>	Définit la classe <i>autosprintf</i> qui rend les routines de sortie formatée C utilisables dans les programmes C++ pour utiliser les chaînes de <code><string></code> et les flux de <code><iostream></code>
<code>libgettextlib</code>	Une bibliothèque privée contenant les routines communes utilisées par les nombreux programmes gettext. Elles ne sont pas faites pour une utilisation généralisçe
<code>libgettextpo</code>	Utilisé pour écrire les programmes spécialisés qui s'occupent des fichiers <code>.po</code> . Cette bibliothèque est utilisée lorsque les applications standards livrées avec Gettext ne vont pas suffire (comme msgcomm , msgcmp , msgattrib et msgen)
<code>libgettextsrc</code>	Une bibliothèque privée contenant les routines communes utilisées par les nombreux programmes gettext. Elles ne sont pas destinées à une utilisation générale
<code>preloadable_libintl.so</code>	Une bibliothèque prévue pour être utilisée par LD_PRELOAD, qui aide libintl à enregistrer des messages non traduits.

10.62. Grep-2.14

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

10.62.1. Installation de Grep

Préparez la compilation de Grep :

```
CC="gcc ${BUILD64}" ./configure --prefix=/usr \  
--bindir=/bin
```

Compilez le paquet :

```
make
```

Pour tester les résultats, exécutez : **make check**.

Installez le paquet :

```
make install
```

10.62.2. 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 |

10.63. Groff-1.21

Le paquet Groff contient des programmes de formatage de texte.

10.63.1. Installation de Groff

Groff s'attend à ce que la variable d'environnement `PAGE` contienne la taille de papier par défaut. Pour des utilisateurs qui vivent aux États-Unis, `PAGE=letter` est approprié. Sinon, il se peut que `PAGE=A4` convienne mieux.

Préparez la compilation de Groff :

```
PAGE=[paper_size] CC="gcc ${BUILD64}" \
CXX="g++ ${BUILD64}" ./configure --prefix=/usr --libdir=/usr/lib64
```

Compilez le paquet :

```
make
```

Ce paquet ne fournit pas de suite de tests.

Installez le paquet :

```
make install
```

Quelques programmes de documentation, comme **xman**, ne fonctionnent pas correctement sans les liens symboliques suivants :

```
ln -sv soelim /usr/bin/zsoelim
ln -sv eqn /usr/bin/geqn
ln -sv tbl /usr/bin/gtbl
```

10.63.2. 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, pre-grohtml, precon, refer, roff2dvi, roff2html, roff2pdf, roff2ps, roff2text, roff2x, soelim, tbl, tfmtodit, troff et zsoelim (lien vers soelim)

Répertoires installés: /usr/lib/groff, /usr/share/doc/groff-1.21, /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 contenues 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 un diagramme grap en image bitmap découpée
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 <code>-e</code> , <code>-man</code> , <code>-me</code> , <code>-mm</code> , <code>-ms</code> , <code>-p</code> , <code>-s</code> , et <code>-t</code> 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 (<i>American Standard Code for Information Interchange</i>)
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 <code>.pfb</code>
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 d'entrée en 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 <code>[</code> et <code>]</code> interprétées comme des citations, et les lignes entre <code>.R1</code> et <code>.R2</code> interprétées comme des commandes sur la façon de gérer les citations
roff2dvi	Transforme des fichiers roff dans d'autres formats

roff2html	Transforme des fichiers roff dans d'autres formats
roff2pdf	Transforme des fichiers roff dans d'autres formats
roff2ps	Transforme des fichiers roff dans d'autres formats
roff2text	Transforme des fichiers roff dans d'autres formats
roff2x	Transforme des fichiers roff dans d'autres formats
soelim	Lit des fichiers et remplace les lignes de la forme <i>file</i>
tbl	Compile les descriptions des tables contenues 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
zsoelim	Un lien vers soelim

10.64. Less-451

Le paquet Less contient un visualisateur de fichiers texte.

10.64.1. Installation de Less

Préparez la compilation de Less :

```
CC="gcc ${BUILD64}" ./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 ne fournit pas de suite de tests.

Installez le paquet :

```
make install
```

Déplacez **less** vers /bin :

```
mv -v /usr/bin/less /bin
```

10.64.2. 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

10.65. Gzip-1.5

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

10.65.1. Installation de Gzip

Préparez la compilation de Gzip :

```
CC="gcc ${BUILD64}" ./configure --prefix=/usr --bindir=/bin
```

Compilez le paquet :

```
make
```

Pour tester les résultats, exécutez : **make check**.

Installez le paquet :

```
make install
```

Maintenant, nous allons déplacer certains outils vers `/usr/bin` pour satisfaire la convention FHS :

```
mv -v /bin/z{egrep,cmp,diff,fgrep,force,grep,less,more,new} /usr/bin
```

10.65.2. 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	Comprime 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	Exécute cmp sur des fichiers compressés avec gzip
zdiff	Exécute diff sur des fichiers compressés avec gzip
zegrep	Exécute egrep sur des fichiers compressés avec gzip
zfgrep	Exécute fgrep sur des fichiers compressés avec gzip
zforce	Force une extension <code>.gz</code> 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	Exécute grep sur des fichiers compressés avec gzip
zless	Exécute less sur des fichiers compressés avec gzip
zmore	Exécute more sur des fichiers compressés avec gzip
znew	Convertit les fichiers formatés avec compress au format gzip — de <code>.Z</code> vers <code>.gz</code>

10.66. IPutils-s20101006

Le paquet IPutils contient des programmes pour du réseau de base.

10.66.1. Installation de IPutils

IPutils a divers problèmes gérés par le correctif suivant :

```
patch -Np1 -i ../iputils-s20101006-fixes-1.patch
```

Le correctif suivant contient la documentation pré-générée pour IPutils :

```
patch -Np1 -i ../iputils-s20101006-doc-1.patch
```

Compilez le paquet :

```
make CC="gcc ${BUILD64}" IPV4_TARGETS="tracepath ping clockdiff rdisc" \
    IPV6_TARGETS="tracepath6 traceroute6"
```

Ce paquet est fourni sans suite de tests.

Installez le paquet :

```
install -v -m755 ping /bin
install -v -m755 clockdiff /usr/bin
install -v -m755 rdisc /usr/bin
install -v -m755 tracepath /usr/bin
install -v -m755 trace{path,route}6 /usr/bin
install -v -m644 doc/*.8 /usr/share/man/man8
```

10.66.2. Contenu de iputils

Programmes installés: clockdiff, ping, rdisc, tracepath, tracepath6, and traceroute6

Descriptions courtes

clockdiff	Mesure la différence d'heures entre des machines
ping	Envoie des paquets echo-request et affiche le temps mis pour que la réponse arrive. C'est la version IPV4.
rdisc	Démon de découverte du routeur réseau
tracepath	Indique le chemin vers une machine du réseau en montrant le MTU tous le long du chemin. C'est la version IPV4.
tracepath6, tracepath6	Indique le chemin vers une machine du réseau en montrant le MTU tous le long du chemin. C'est la version IPV6.
traceroute6	Indique le chemin vers une machine du réseau sur réseau IPV6

10.67. Kbd-1.15.3

Le paquet Kbd contient les fichiers de plan de codage et des outils pour le clavier.

10.67.1. Installation de Kbd

Appliquez le correctif suivant pour corriger une coquille dans es.po :

```
patch -Np1 -i ../kbd-1.15.3-es.po_fix-1.patch
```

Préparez la compilation de Kbd :

```
CC="gcc ${BUILD64}" ./configure --prefix=/usr
```

Compilez le paquet :

```
make CC="gcc ${BUILD64}"
```

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

Installez le paquet :

```
make install
```

Certains programmes de Kbd sont utilisés par des scripts du paquet CLFS-Bootscripts. Comme il se peut que /usr ne soit pas disponibles lors des premières étapes du démarrage, ces binaires doivent être sur la partition racine :

```
mv -v /usr/bin/{kbd_mode,dumpkeys,loadkeys,openvt,setfont,setvtrgb} /bin
```

10.67.2. Contenu de Kbd

Programmes installés:	chvt, deallocvt, dumpkeys, fgconsole, getkeycodes, kbinfo, kbd_mode, kbdrate, loadkeys, loadunimap, mapscrn, openvt, psfaddtable (lien vers psfxtable), psfgettable (lien vers psfxtable), psfstriptime (lien vers psfxtable), psfxtable, resizecons, setfont, setkeycodes, settled, setmetamode, setvtrgb, showconsolefont, showkey, unicode_start, et unicode_stop
Répertoires installés:	/usr/share/consolefonts, /usr/share/consoletrans, /usr/share/keymaps, /usr/share/unimaps

Descriptions courtes

chvt	Change le terminal virtuel en avant plan
deallocvt	Désalloue les terminaux virtuels inutilisés
dumpkeys	
fgconsole	Affiche le numéro du terminal virtuel actif
getkeycodes	Affiche la table de correspondance des « scancode » avec les « keycode »
kbinfo	Récupère des informations concernant la 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
psfstriptide	Un lien vers psfxtable
psfxtable	Gère les tables de caractères Unicode pour les polices de la console
resizecons	Change l'idée du noyau sur la taille de la console
setfont	Modifie les polices EGA/VGA (<i>Enhanced Graphic Adapter-Video Graphics Array</i> 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
setvtrgb	Règle les couleurs RGB du terminal virtuel
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. Ne l'utilisez pas sur CLFS 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

10.68. Make-3.82

Le paquet Make contient un programme pour compiler des paquets.

10.68.1. Installation de Make

Préparez la compilation de Make :

```
CC="gcc ${BUILD64}" ./configure --prefix=/usr
```

Compilez le paquet :

```
make
```

Pour tester les résultats, exécutez : **make check**.

Installez le paquet :

```
make install
```

10.68.2. Contenu de Make

Programme installé: **make**

Descriptions courtes

make Détermine automatiquement quelles parties d'un paquet doivent être (re)compilées. Puis, il exécute les commandes adéquates

10.69. Outils XZ Utils-5.0.4 32 bits

Le paquet XZ-Utils contient des programmes pour compresser et décompresser des fichiers. La compression de fichiers texte avec **XZ-Utils** donne un pourcentage de compression bien meilleur qu'avec le **gzip** traditionnel.

10.69.1. Installation de XZ Utils

Préparez la compilation de XZ-Utils :

```
CC="gcc ${BUILD32}" ./configure --prefix=/usr --libdir=/lib
```

Compilez le paquet :

```
make
```

Pour tester les résultats, lancez : **make check**.

Installez les programmes :

```
make pkgconfigdir=/usr/lib/pkgconfig install
```

Déplacez le binaire xz et plusieurs liens symboliques qui pointent vers lui dans le répertoire /bin :

```
mv -v /usr/bin/{xz,lzma,lzcat,unlzma,unxz,xzcat} /bin
```

Déplacez les bibliothèques statiques au bon endroit :

```
mv -v /lib/liblzma.a /usr/lib
```

Les détails sur ce paquet sont disponibles dans Section 10.70.2, « Contenu de XZ-Utils. »

10.70. XZ Utils-5.0.4 64 bits

Le paquet XZ-Utils contient des programmes pour compresser et décompresser des fichiers. La compression de fichiers texte avec **XZ-Utils** donne un pourcentage de compression bien meilleur qu'avec le **gzip** traditionnel.

10.70.1. Installation de XZ-Utils

Préparez la compilation de XZ-Utils :

```
CC="gcc ${BUILD64}" ./configure --prefix=/usr --libdir=/lib64
```

Compilez le paquet :

```
make
```

Pour tester les résultats, lancez : **make check**.

Installez les programmes :

```
make pkgconfigdir=/usr/lib64/pkgconfig install
```

Déplacez le binaire xz et plusieurs liens symboliques qui pointent vers lui dans le répertoire /bin :

```
mv -v /usr/bin/{xz,lzma,lzcat,unlzma,unxz,xzcat} /bin
```

Déplacez les bibliothèques statiques au bon endroit :

10.70.2. Contenu de XZ-Utils

Programmes installés:	lzcat (link to xz), lzcmp (lien vers lzdiff), lzdiff, lzegrep (lien vers lzgrep), lzfgrep (lien vers lzgrep), lzgrep, lzless (lien vers lzmore), lzma (lien vers xz), lzmadec, lzmore, unlzma (lien vers xz), unxz (lien vers xz), xz, xzcat (lien vers xz) et xzdec
Bibliothèques installées:	liblzma.[a,so]
Répertoires installés:	/usr/include/lzma, /usr/share/doc/xz

Descriptions courtes

lzcat	Décompresse des fichiers LZMA et xz
lzcmp	Compare des fichiers compressés avec lzma
lzdiff	Compare des fichiers compressés avec lzma
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
lzma	Comprime des fichiers lzma
lzmadec	Décompresse des fichiers lzma
lzmore	Lance more sur des fichiers lzma
unlzma	Décompresse des fichiers lzma
unxz	Décompresse des fichiers xz
xz	Crée des fichiers compressés xz
xzcat	Décompresse des fichiers xz

xzdec Décompresse vers la sortie standard

liblzma La bibliothèque LZMA

10.71. Man-1.6g

Le paquet Man contient des programmes pour trouver et voir des pages de manuel.

10.71.1. Installation de Man

Ce correctif ajoute le support de l'internationalisation :

```
patch -Np1 -i ../man-1.6g-i18n-1.patch
```

Il faut effectuer quelques ajustements aux sources de Man.

D'abord, une substitution **sed** est nécessaire pour ajouter l'option `-R` à la variable `PAGER` afin que les séquences d'échappement soient correctement gérées par Less :

```
sed -i 's@-is@&R@g' configure
```

Deux autres substitutions **sed** commentent les lignes « `MANPATH /usr/man` » et « `MANPATH /usr/local/man` » dans le fichier `man.conf` pour empêcher des résultats redondants lors de l'utilisation de programmes tels que **whatis** :

```
sed -i 's@MANPATH./usr/man@#&@g' src/man.conf.in
sed -i 's@MANPATH./usr/local/man@#&@g' src/man.conf.in
```

Préparez la compilation de Man :

```
CC="gcc ${BUILD64}" ./configure --confdir=/etc
```

Voici la signification des options de configure :

`--confdir=/etc`

Ceci dit au programme **man** de chercher le fichier de configuration `man.conf` dans le répertoire `/etc`.

Compilez le paquet :

```
make
```

Ce paquet ne fournit pas de suite de tests.

Installez le paquet :

```
make install
```



Remarque

Si vous allez travailler sur un terminal qui ne supporte pas les attributs de texte comme la couleur ou le gras, vous pouvez désactiver les séquences d'échappement Select Graphic Rendition (SGR) en éditant le fichier `man.conf` et en ajoutant l'option `-c` à la variable `NROFF`. Si vous utilisez plusieurs types de terminal pour un ordinateur, il peut être préférable d'ajouter de manière sélective la variable d'environnement `GROFF_NO_SGR` pour les terminaux qui ne supportent pas SGR.

Si l'encodage de la locale utilise les caractères 8 bits, cherchez la ligne commençant par « `NROFF` » dans `/etc/man.conf` et vérifiez qu'elle correspond à ce qui suit :

```
NROFF /usr/bin/nroff -Tlatin1 -mandoc
```

Remarquez que vous devriez utiliser « `latin1` » même si ce n'est pas l'encodage de la locale. La raison à cela est que, selon la spécification, **groff** n'attribue aucun sens aux caractères différents de *International Organization for Standards* (ISO) 8859-1 sans quelques codes d'échappement bizarres. Lorsqu'il formate des pages de man, **groff**

pense qu'elles sont en encodage ISO 8859-1 et ce paramètre `-Tlatin1` dit à **groff** d'utiliser le même encodage pour la sortie. Comme **groff** ne fait pas de recodage des caractères d'entrée, le résultat formaté est vraiment dans le même encodage que l'entrée et ainsi, il est utilisable comme l'entrée pour un pager.

Cela ne résout pas le problème du programme **man2dvi** qui ne fonctionne pas pour les pages de man non localisées en locales ISO 8859-1. En outre, il ne fonctionne pas avec les encodages multioctets. Le premier problème n'a aucune solution actuellement. Le second problème ne nous concerne pas car l'installation de CLFS ne supporte pas les encodages multioctets.

10.71.2. Contenu de Man

Programmes installés: `apropos`, `makewhatis`, `man`, `man2dvi`, `man2html` et `whatis`

Descriptions courtes

apropos	Cherche la base de données whatis et affiche les descriptions courtes des commandes système qui contiennent une chaîne donnée
makewhatis	Construit la base de données whatis ; il lit toutes les pages de man dans <code>MANPATH</code> et écrit le nom et une courte description dans la base de données whatis pour chaque page
man	Formate et affiche la page de manuel en ligne demandée
man2dvi	Convertit une page de manuel au format dvi
man2html	Convertit une page de manuel en HTML
whatis	Cherche la base de données whatis et affiche les descriptions courtes des commandes système qui contiennent le mot-clé donné

10.72. Kmod-10 32 Bit Libraries

Le paquet Kmod contient des programmes pour charger, insérer et supprimer des modules du noyau pour Linux. Kmod remplace le paquet Module-Init-tools.

10.72.1. Installation de Kmod

Préparez la compilation de Kmod :

```
PKG_CONFIG_PATH=${PKG_CONFIG_PATH32} CC="gcc ${BUILD32}" \
./configure --prefix=/usr \
--bindir=/bin --sysconfdir=/etc \
--with-rootlibdir=/lib --libdir=/usr/lib \
--with-zlib --with-xz
```

Voici la signification des options de configure :

--with-rootlibdir=/lib

Emplacement d'installation des bibliothèques partagées.

--with-zlib --with-xz

Ceci permet au paquet Kmod de gérer les modules du noyau compressés avec zlib et XZ.

Compile the package:

```
make
```

Pour tester les résultats, lancez : **make check**

Installez le paquet :

```
make install
```

Les détails sur ce paquet sont disponibles dans Section 10.73.2, « Contenu de Kmod. »

10.73. Kmod-10 64 Bit

Le paquet Kmod contient des programmes pour charger, insérer et supprimer des modules du noyau pour Linux. Kmod remplace le paquet Module-Init-tools.

10.73.1. Installation de Kmod

Préparz la compilation de Kmod :

```
PKG_CONFIG_PATH=${PKG_CONFIG_PATH64} CC="gcc ${BUILD64}" \
./configure --prefix=/usr \
--bindir=/bin --sysconfdir=/etc \
--with-rootlibdir=/lib64 --libdir=/usr/lib64 \
--with-zlib --with-xz
```

Voici la signification des options de configure :

`--with-rootlibdir=/lib`

Emplacement d'installation des bibliothèques partagées.

`--with-zlib --with-xz`

Ceci permet au paquet Kmod de gérer les modules du noyau compressés avec zlib et XZ.

Compile the package:

```
make
```

Pour tester les résultats, lancez : **make check**

Installez le paquet :

```
make install
```

Créez des liens symboliques pour les programmes qui cherchent Module-Init-Tools.

```
ln -sv kmod /bin/lsmmod
ln -sv ../bin/kmod /sbin/depmod
ln -sv ../bin/kmod /sbin/insmod
ln -sv ../bin/kmod /sbin/modprobe
ln -sv ../bin/kmod /sbin/modinfo
ln -sv ../bin/kmod /sbin/rmmmod
```

10.73.2. Contenu de Kmod

Programmes installés: depmod, insmod, kmod, lsmmod, modinfo, modprobe et rmmmod

Descriptions courtes

depmod	Crée un fichier de dépendances basé sur les symboles qu'il trouve dans le jeu de modules existant ; ce fichier de dépendances est utilisé par modprobe pour charger automatiquement les modules requis
insmod	Installe un module chargeable dans le noyau en fonction
kmod	Charge et décharge des modules du noyau
lsmmod	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

10.74. 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**.

10.74.1. Installation de Patch

Préparez la compilation de Patch :

```
CC="gcc ${BUILD64}" ./configure --prefix=/usr
```

Compilez le paquet :

```
make
```

Pour tester les résultats, lancez : **make check**.

Installez le paquet :

```
make install
```

10.74.2. Contenu de Patch

Programme installé: patch

Descriptions courtes

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.

10.75. Psmisc-22.20

Le paquet Psmisc contient des programmes pour afficher des informations sur les processus en cours d'exécution.

10.75.1. Installation de Psmisc

Préparez la compilation de Psmisc :

```
CC="gcc ${BUILD64}" ./configure --prefix=/usr \
--exec-prefix=""
```

Voici la signification de l'option de configure :

```
--exec-prefix=""
```

Ceci nous assure que les binaires de Psmisc sont installés dans `/bin` au lieu de `/usr/bin`. D'après le FHS, il s'agit du bon emplacement car certains binaires de Psmisc sont utilisés dans le paquet CLFS-Bootscripts.

Compilez le paquet :

```
make
```

Ce paquet ne fournit pas de suite de tests.

Installez le paquet :

```
make install
```

Il n'existe aucune raison pour que les programmes **pstree** et **pstree.x11** résident dans `/bin`. Du coup, déplacez-les dans `/usr/bin` :

```
mv -v /bin/pstree* /usr/bin
```

Par défaut, le programme **pidof** de Psmisc n'est pas installé. Généralement, ce n'est pas un problème car le paquet Sysvinit installe une meilleure version de **pidof**. Mais si Sysvinit ne sera pas utilisé, terminez l'installation de Psmisc en créant le lien symbolique suivant :

```
ln -sv killall /bin/pidof
```

10.75.2. 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	Recherche les descripteurs de fichiers des processus en cours d'exécution
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

10.76. Libestr-0.1.0 32 Bit Libraries

Le paquet Libestr est une bibliothèque de chaînes essentielles.

10.76.1. Installation de Libestr

Préparez la compilation de Libestr :

```
CC="gcc ${BUILD32}" ./configure --prefix=/usr
```

Compilez le paquet :

```
make
```

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

Installez le paquet :

```
make install
```

Les détails sur ce paquet sont disponibles dans Section 10.77.2, « Contenu de Libestr. »

10.77. Libestr-0.1.0 64 Bit

Le paquet Libestr est une bibliothèque de chaînes essentielles.

10.77.1. Installation de Libestr

Préparez la compilation de Libestr :

```
CC="gcc ${BUILD64}" ./configure --prefix=/usr \  
--libdir=/usr/lib64
```

Compilez le paquet :

```
make
```

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

Installez le paquet :

```
make install
```

10.77.2. Contenu de Libestr

Bibliothèques installées: libestr.[a,so]

Descriptions courtes

`libestr` contient des fonctions d'aide pour des fonctions de chaîne

10.78. Libee-0.4.1 32 Bit Libraries

Le paquet Libee est une bibliothèque d'expression d'événements.

10.78.1. Installation de Libee

Préparez la compilation de Libee :

```
CC="gcc ${BUILD32}" PKG_CONFIG_PATH="${PKG_CONFIG_PATH32}" \  
./configure --prefix=/usr
```

Compilez le paquet :



Remarque

Libee ne pourra pas se construire si vous utilisez plusieurs tâches avec make. Mettez "-j 1" dans la commande make suivante :

```
make
```

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

Installez le paquet :

```
make install
```

Les détails sur ce paquet sont disponibles dans Section 10.79.2, « Contenu de Libee. »

10.79. Libee-0.4.1 64 Bit

Le paquet Libee est une bibliothèque d'expression d'événements.

10.79.1. Installation de Libee

Préparez la compilation de Libee :

```
CC="gcc ${BUILD64}" PKG_CONFIG_PATH="${PKG_CONFIG_PATH64}" \
./configure --prefix=/usr \
--libdir=/usr/lib64
```

Compilez le paquet :



Remarque

Libee ne pourra pas se construire si vous utilisez plusieurs tâches avec make. Mettez "-j 1" dans la commande make suivante :

```
make
```

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

Installez le paquet :

```
make install
```

10.79.2. Contenu de Libee

Programme installé:	libee-convert
Bibliothèques installées:	libee.[a,so]
Répertoire installé:	/usr/include/libee

Descriptions courtes

libee-convert	todo
libee	est la bibliothèque d'expression d'événements

10.80. Rsyslog-6.2.2

Le paquet rsyslog contient des programmes pour les messages du système de fichier journal, tels que ceux fournis par le noyau lorsque des choses inhabituelles se produisent.

10.80.1. Installation de Rsyslog

Préparez la compilation de Rsyslog :

```
CC="gcc ${BUILD64}" PKG_CONFIG_PATH="${PKG_CONFIG_PATH64}" \  
./configure --prefix=/usr --libdir=/usr/lib64
```

Compilez le paquet :

```
make
```

Pour tester les résultats, lancez : **make check**.

Installez le paquet :

```
make install
```

Créez un répertoire pour les expansion snippets :

```
install -dv /etc/rsyslog.d
```

```

# Support pour le système journal du système local
$ModLoad imuxsock.so

# Support pour la journalisation du noyau
$ModLoad imklog.so

#####
# Options globales

# Utiliser le format traditionnel d'horodateur.
$ActionFileDefaultTemplate RSYSLOG_TraditionalFileFormat

# Réglage des droits par défaut pour tous les fichiers journaux
$FileOwner root
$FileGroup root
$FileCreateMode 0640
$DirCreateMode 0755

# Fournit la réception UDP
$ModLoad imudp
$UDPServerRun 514

# Désactive la répétition des entrées
$RepeatedMsgReduction on

#####
# Inclut les snippets de config de Rsyslog

$IncludeConfig /etc/rsyslog.d/*.conf

#####
# Fichiers journaux standard

auth,authpriv.*    /var/log/auth.log
*.*;auth,authpriv.none  -/var/log/syslog
daemon.*    -/var/log/daemon.log
kern.*      -/var/log/kern.log
lpr.*       -/var/log/lpr.log
mail.*      -/var/log/mail.log
user.*      -/var/log/user.log

# Récupérer tous les journaux
*.=debug;\
auth,authpriv.none;\
news.none;mail.none -/var/log/debug
*.=info;*.=notice;*.=warn;\
auth,authpriv.none;\
cron,daemon.none;\
mail,news.none  -/var/log/messages

# On montre les urgences à tout le monde
*.emerg        *

# Fin de /etc/rsyslog.conf
EOF

```

10.80.3. Contenu de rsyslog

Programmes installés: rsyslogd
Répertoire installé: /usr/lib/rsyslog

Descriptions courtes

rsyslogd Enregistre les messages que le système donne à journaliser. Tout message enregistré contient au moins une date et un nom d'hôte et en principe également le nom du programme, mais cela dépend de lu niveau de vigilance dont vous avez dit au démon de journal de faire preuve.

10.81. Sysvinit-2.88dsf

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

10.81.1. Installation de Sysvinit

Appliquez un sed qui désactive la construction et l'installation de sulogin, mountpoint, wall et utmpdump vu qu'ils sont fournis par util-linux :

```
sed -i -e 's/\ sulogin[^\ ]*//' \
    -e '/utmpdump/d' -e '/mountpoint/d' src/Makefile
```

Compilez le paquet :

```
make -C src clobber
make -C src CC="gcc ${BUILD64}"
```

Installez le paquet :

```
make -C src install
```

10.81.2. Configurer Sysvinit

Créez un nouveau fichier /etc/inittab en lançant ce qui suit :

```
cat > /etc/inittab << "EOF"
# Début de /etc/inittab

id:3:initdefault:

si::sysinit:/etc/rc.d/init.d/rc sysinit

10:0:wait:/etc/rc.d/init.d/rc 0
11:S1:wait:/etc/rc.d/init.d/rc 1
12:2:wait:/etc/rc.d/init.d/rc 2
13:3:wait:/etc/rc.d/init.d/rc 3
14:4:wait:/etc/rc.d/init.d/rc 4
15:5:wait:/etc/rc.d/init.d/rc 5
16:6:wait:/etc/rc.d/init.d/rc 6

ca:12345:ctrlaltdel:/sbin/shutdown -t1 -a -r now

su:S016:once:/sbin/sulogin

EOF
```

La commande suivante ajoute les terminaux virtuels standards à `/etc/inittab`. Si votre système n'a qu'une console série, passez la commande suivante :

```
cat >> /etc/inittab << "EOF"
1:2345:respawn:/sbin/agetty -I '\033(K' tty1 9600
2:2345:respawn:/sbin/agetty -I '\033(K' tty2 9600
3:2345:respawn:/sbin/agetty -I '\033(K' tty3 9600
4:2345:respawn:/sbin/agetty -I '\033(K' tty4 9600
5:2345:respawn:/sbin/agetty -I '\033(K' tty5 9600
6:2345:respawn:/sbin/agetty -I '\033(K' tty6 9600

EOF
```

Si votre système a une console série, lancez la commande suivante pour ajouter l'entrée à `/etc/inittab` :

```
cat >> /etc/inittab << "EOF"
c0:12345:respawn:/sbin/agetty 115200 ttyS0 vt100

EOF
```

Enfin, ajoutez la ligne de fin à `/etc/inittab` :

```
cat >> /etc/inittab << "EOF"
# Fin de /etc/inittab
EOF
```

L'option `-I '\033(K'` dit à **agetty** d'envoyer cette séquence d'échappement au terminal avant de faire quoique ce soit. Cette séquence d'échappement bascule l'encodage de la console défini par l'utilisateur, qui peut être modifié en lançant le programme **setfont**. Le script de démarrage **console** du paquet CLFS-Bootscripts appelle le programme **setfont** pendant le démarrage du système. L'envoi de cette séquence d'échappement est nécessaire pour les gens qui utilisent des polices d'écran non ISO 8859-1, mais il n'affecte pas les anglophones d'origine.

10.81.3. 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 et telinit (lien vers init)

Descriptions courtes

bootlogd	Trace les messages de démarrage dans le journal
fstab-decode	Lance une commande avec des arguments encodés fstab
halt	Lance normalement shutdown avec l'option <code>-h</code> , 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 <code>/var/log/wtmp</code> 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 <code>/var/log/wtmp</code> . 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 <code>/var/log/btmp</code>
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 <code>/var/run/utmp</code>
shutdown	Arrête proprement le système en le signalant à tous les processus et à tous les utilisateurs connectés
telinit	Indique à init dans quel niveau d'exécution entrer

10.82. Tar-1.26

Le paquet Tar contient un programme d'archivage.

10.82.1. Installation de Tar

Le correctif suivant ajoute une page de man pour **tar** :

```
patch -Np1 -i ../tar-1.26-man-1.patch
```

Préparez la compilation de Tar :

```
FORCE_UNSAFE_CONFIGURE=1 CC="gcc ${BUILD64}" \  
./configure --prefix=/usr \  
--bindir=/bin --libexecdir=/usr/sbin
```

Compilez le paquet :

```
make
```

Pour tester les résultats, exécutez : **make check**.

Installez le paquet :

```
make install
```

10.82.2. Contenu de Tar

Programmes installés: rmt and tar

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

10.83. Texinfo-4.13a

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

10.83.1. Installation de Texinfo

Le correctif suivant ajoute le support pour les nouveaux outils de compression comme XZ Utils :

```
patch -Np1 -i ../texinfo-4.13a-new_compressors-1.patch
```

Préparez la compilation de Texinfo :

```
CC="gcc ${BUILD64}" ./configure --prefix=/usr
```

Compilez le paquet :

```
make
```

Pour tester les résultats, exécutez : **make check**.

Installez le paquet :

```
make install
```

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 :

```
pushd /usr/share/info
rm dir
for f in *
do install-info $f dir 2>/dev/null
done
popd
```

10.83.2. Contenu de Texinfo

Programmes installés: info, infokey, install-info, makeinfo, pdftexi2dvi, 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
pdftexi2dvi	Script shell qui lance texi2dvi --pdf
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 (<i>Portable Document Format</i>)
texindex	Utilisé pour trier les fichiers d'index de Texinfo

10.84. Bibliothèques Udev-182 32 bits

Le paquet Udev contient des programmes pour créer dynamiquement des nœuds périphériques.

10.84.1. Installation de Udev

Préparez la compilation d'Udev :

```
PKG_CONFIG_PATH=${PKG_CONFIG_PATH32} CC="gcc ${BUILD32}" \  
./configure --prefix=/usr --sysconfdir=/etc --with-rootprefix="" \  
--libexecdir=/lib --bindir=/sbin \  
--with-usb-ids-path=no --with-pci-ids-path=no \  
--enable-rule_generator --disable-introspection --disable-keymap \  
--disable-gudev
```

Compilez le paquet :

```
make
```

Pour tester les résultats, lancez : **make check**.

Installez le paquet :

```
make install
```

Les détails sur ce paquet sont disponibles dans Section 10.85.2, « Contenu de Udev. »

10.85. Udev-182 64 bits

Le paquet Udev contient des programmes pour créer dynamiquement des nœuds périphériques.

10.85.1. Installation de Udev

Préparez la compilation d'Udev :

```
PKG_CONFIG_PATH=${PKG_CONFIG_PATH64} CC="gcc ${BUILD64}" \
  --with-usb-ids-path=no --with-pci-ids-path=no \
  ./configure --prefix=/usr --sysconfdir=/etc --with-rootprefix="" \
  --libexecdir=/lib64 --libdir=/usr/lib64 --bindir=/sbin \
  --enable-rule_generator --disable-introspection --disable-keymap \
  --disable-gudev --with-firmware-path=/lib/firmware
```

Compilez le paquet :

```
make
```

Pour tester les résultats, lancez : **make check**.

Installez le paquet :

```
make install
```

Créez un répertoire pour le stockage des firmware qui peuvent être chargés par **udev** :

```
install -dv /lib/firmware
```

10.85.2. Contenu de Udev

Programmes installés:	ata_id, cdrom_id, collect, create_floppy_devices, edd_id, firmware.sh, fstab_import, path_id, scsi_id, udevadm, udevd, usb_id, v4l_id, write_cd_rules, write_net_rules
Bibliothèques installées:	libudev
Répertoires installés:	/etc/udev, /lib/firmware, /lib/udev, /usr/share/doc/udev

Descriptions courtes

udevadm	Contrôle le comportement d'Udev pendant son exécution, interroge les événements du noyau, gère la queue d'événements et fournit un débogage simple.
udevadm control	Configure un certain nombre d'options pour le démon udevd existant, telles que le niveau de traçage (lien symbolique vers udevadm)
udevd	Un démon qui réorganise les événements à chaud avant de les soumettre à udev , évitant ainsi divers types de conditions
udevinfo	Autorise les utilisateurs à interroger la base de données udev pour des informations sur un périphérique actuellement présent sur le système ; il fournit aussi une manière d'interroger un périphérique dans l'arborescence <code>sysfs</code> pour aider à créer des règles udev (lien symbolique vers udevadm)
udevadm monitor	Affiche l'événement reçu depuis le noyau et l'événement qu' udev crée après avoir effectué la règle
udevsettle	Regarde la queue d'événements Udev et quitte si tous les événements actuels ont été gérés (lien symbolique vers udevadm)

udevadm test	Simule une exécution d' udev pour le périphérique donné et affiche le nom du nœd que le vrai udev aurait créé ou le nom de l'interface réseau renommée
udevadm trigger	Parcourt l'arborescence de sysfs à la recherche de périphériques qui doivent être ajoutés au système.
ata_id	Fournit Udev avec une chaîne unique et des informations supplémentaires (uuid, label) pour un disque ATA
cdrom_id	Affiche les possibilités d'un lecteur CD-ROM ou DVD-ROM
create_floppy_devices	Crée tous les périphériques amovibles possibles basés sur le type CMOS
dasd_id	Lit le label depuis un bloc de périphérique s390.
edd_id	Identifie des lecteurs de disque x86 pour les appels <i>Enhanced Disk Drive</i> .
firmware.sh	Script pour charger le firmware d'un périphérique
path_id	Fournit le chemin de matériel unique le plus court possible vers un périphérique
scsi_id	Récupère ou génère un identifieur SCSI unique.
usb_id	Identifie un bloc de périphérique USB.
v4l_id	DESCRIPTION REQUIRED
write_cd_rules	DESCRIPTION REQUIRED
write_net_rules	DESCRIPTION REQUIRED
<code>/lib/udev</code>	Contient les programmes d'aide de udev et les périphériques statiques qui sont copiés dans <code>/dev</code> après le démarrage.
<code>/etc/udev</code>	Contient les fichiers de configuration udev , les droits de périphérique et les règles pour le nommage des périphériques

10.86. Vim-7.3

Le paquet Vim contient un puissant éditeur de texte.

10.86.1. Installation de Vim



Alternatives à Vim

Si vous préférez un autre éditeur—comme Emacs, Joe, ou Nano—merci de vous référer à http://cblfs.cross-lfs.org/index.php/Category:Text_Editors pour des instructions d'installation.

Le correctif suivant incorpore toutes les mises à jour de la branche 7.3 issue des développeurs de Vim :

```
patch -Np1 -i ../vim-7.3-branch_update-6.patch
```

Modifiez l'emplacement par défaut du fichier de configuration vimrc vers /etc :

```
echo '#define SYS_VIMRC_FILE "/etc/vimrc"' >> src/feature.h
```

Préparez la compilation de Vim :

```
CC="gcc ${BUILD64}" CXX="g++ ${BUILD64}" \  
./configure --prefix=/usr \  
--enable-multibyte
```

Voici la signification des options 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 ensemble de caractères par défaut.

Compilez le paquet :

```
make
```

Pour tester les résultats, exécutez : **make test**. Néanmoins, cette suite de tests affiche beaucoup de données binaires à l'écran, ce qui peut provoquer des problèmes avec les paramètres du terminal actuel. Vous pouvez résoudre cela en redirigeant la sortie vers un fichier journal

Installez le paquet :

```
make install
```

Beaucoup d'utilisateurs sont habitués à utiliser **vi** au lieu de **vim**. Certains programmes comme **vigr** et **vipw** utilisent aussi **vi**. Créez un lien symbolique pour permettre l'exécution de **vim** lorsque les utilisateurs entrent habituellement **vi** et pour permettre aux programmes qui utilisent **vi** de fonctionner :

```
ln -sv vim /usr/bin/vi
```

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 CLFS, 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 CBLFS sur <http://cblfs.cross-lfs.org/index.php/Vim>.

10.86.2. Configurer Vim

Par défaut, **vim** est lancé en mode compatible vi. Ceci pourrait être nouveau pour les personnes qui ont utilisé d'autres éditeurs dans le passé. Le paramètre « *nocompatible* » est inclus ci-dessous pour surligner le fait qu'un nouveau comportement est en cours d'utilisation. Il rappelle aussi à ceux qui voudraient le changer en mode « compatible » qu'il devrait être le premier paramètre dans le fichier de configuration. Ceci est nécessaire car il modifie d'autres paramètres et la surcharge doit survenir après ce paramètre. Créez un fichier de configuration **vim** par défaut en lançant ce qui suit :

```
cat > /etc/vimrc << "EOF"
" Début de /etc/vimrc

set nocompatible
set backspace=2
set ruler
syntax on
if (&term == "iterm") || (&term == "putty")
    set background=dark
endif

" Fin de /etc/vimrc
EOF
```

L'option *set nocompatible* change le comportement de **vim** d'une façon plus utile 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'
```

10.86.3. Contenu de Vim

Programmes installés:	efm_filter.pl, efm_perl.pl, ex (link to vim), less.sh, mve.awk, pltags.pl, ref, rview (link to vim), rvim (link to vim), shtags.pl, tcltags, vi (link to vim), view (link to vim), vim, vim132, vim2html.pl, vimdiff (link to vim), vimm, vimspell.sh, vintutor et xxd
Répertoire installé:	/usr/share/vim

Descriptions courtes

efm_filter.pl	Un filtre pour créer un fichier d'erreurs lisible par vim
efm_perl.pl	Reformatte les messages d'erreur de l'interpréteur Perl pour une utilisation avec le mode « quickfix » de vim

ex	Lance vim en mode ex
less.sh	Un script qui démarre vim avec less.vim
mve.awk	Montre les erreurs de vim
pltags.pl	Crée un fichier de balises pour le code Perl pour une utilisation par vim
ref	Vérifie l'orthographe des arguments
rview	Est une version restreinte de view ; aucune commande shell ne peut être lancée et view ne peut pas être suspendu
rvim	Est une version restreinte de vim ; aucune commande shell ne peut être lancée et vim ne peut pas être suspendu
shtags.pl	Génère un fichier de balises pour les scripts Perl
tcltags	Génère un fichier de balises pour les scripts TCL
view	Lance vim en mode lecture seule
vi	Lien vers vim
vim	Est l'éditeur
vim132	Lance vim avec le mode terminal en 132 colonnes
vim2html.pl	Convertit la documentation Vim en <i>HyperText Markup Language</i> (HTML)
vimdiff	Edite deux ou trois versions d'un fichier avec vim et montre les différences
vimm	Active le modèle d'entrée DEC locator sur un terminal distant
vimspell.sh	Vérifie l'orthographe d'un fichier et génère l'état de la syntaxe qui doit être surlignée dans vim . Ce script exige la vieille commande spell qui n'est fournie ni dans CLFS ni dans CBLFS
vimtutor	Enseigne les touches et les commandes de base de vim
xxd	Crée un hexa du fichier donné ; il peut aussi faire l'inverse et peut donc être utilisé pour corriger du binaire

10.87. GRUB-2.00

Le paquet GRUB contient le *GRand Unified Bootloader*.

10.87.1. Installation de GRUB



Remarque

Si vous aimeriez utiliser un autre chargeur de démarrage, vous pouvez vous rendre à l'adresse suivante pour des chargeurs de démarrage alternatifs et les instructions pour les utiliser. <http://trac.cross-lfs.org/wiki/bootloaders>



Remarque

Ce paquet est connu pour avoir des problèmes quand on change ses drapeaux d'optimisation (y compris les options `-march` et `-mcpu`). Si vous avez défini une variable d'environnement remplaçant les optimisations par défaut, telles que `CFLAGS` et `CXXFLAGS`, désinitialisez-les lors de la construction de GRUB.

Préparez la construction de GRUB :

```
./configure --prefix=/usr \  
--sysconfdir=/etc --disable-werror
```

Compilez le paquet :

```
make
```

Pour tester GRUB vous devez avoir installé QEMU et lancer : **make check**.

Installez le paquet :

```
make install
```

10.87.2. Configurer GRUB

Maintenant que grub est installé, il faut configurer les paramètres par défaut Utilisés pour générer la configuration après qu'on a installé le noyau. Créez ce fichier avec ce qui suit :

```
install -m755 -dv /etc/default
cat > /etc/default/grub << "EOF"
# Begin /etc/default/grub

GRUB_DEFAULT=0
#GRUB_SAVEDEFAULT=true
GRUB_HIDDEN_TIMEOUT=
GRUB_HIDDEN_TIMEOUT_QUIET=false
GRUB_TIMEOUT=10
GRUB_DISTRIBUTOR=Cross-LFS

GRUB_CMDLINE_LINUX=" "
GRUB_CMDLINE_LINUX_DEFAULT=" "

#GRUB_TERMINAL=console
#GRUB_GFXMODE=640x480
#GRUB_GFXPAYLOAD_LINUX=keep

#GRUB_DISABLE_LINUX_UUID=true
#GRUB_DISABLE_LINUX_RECOVERY=true

#GRUB_INIT_TUNE="480 440 1"

#GRUB_DISABLE_OS_PROBER=true

# End /etc/default/grub
EOF
```

Voici la signification des options ci-dessus et les autres valeurs possibles :

```
GRUB_DEFAULT=
    Write Me

GRUB_SAVEDEFAULT=
    Write Me

GRUB_HIDDEN_TIMEOUT=
    Write Me

GRUB_HIDDEN_TIMEOUT_QUIET=
    Write Me

GRUB_TIMEOUT=
    Write Me

GRUB_DISTRIBUTOR=
    Write Me

GRUB_CMDLINE_LINUX=
    Write Me
```

`GRUB_CMDLINE_LINUX_DEFAULT=`

Write Me

`GRUB_TERMINAL=`

Write Me

`GRUB_GFXMODE=`

Write Me

`GRUB_GFXPAYLOAD_LINUX=`

Write Me

`GRUB_DEFAULT=`

Write Me

`GRUB_DISABLE_LINUX_UUID=`

Write Me

`GRUB_DISABLE_LINUX_RECOVERY=`

Write Me

`GRUB_INIT_TUNE=`

Write Me

`GRUB_DISABLE_OS_PROBER=`

Write Me

10.87.3. Contenu de GRUB

Programmes installés: grub, grub-install, grub-md5-crypt, grub-set-default, grub-terminfo, et mbchk
Répertoires installés: /usr/lib/grub, /boot/grub

Descriptions courtes

grub	La ligne de commande de <i>Grand Unified Bootloader</i>
grub-install	Installe GRUB sur le périphérique donné
grub-md5-crypt	Chiffre un mot de passe au format MD5
grub-set-default	Règle l'entrée de démarrage par défaut pour GRUB
grub-terminfo	Génère une commande terminfo à partir d'un nom terminfo ; on peut l'utiliser si on va utiliser un terminal inconnu
mbchk	Vérifie le format d'un noyau multi-amorçage

10.88. À propos des symboles de débogage

Par défaut, la plupart des programmes et des bibliothèques sont compilés 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 routines.

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 Kio
- un binaire `bash` sans les symboles de débogage : 480 Kio
- fichiers de GCC et de Glibc (`/lib`, `/lib64`, `/usr/lib`, et `/usr/lib64`) avec les symboles de débogage : 87 Mo
- Les fichiers de Glibc et GCC sans les symboles de débogage : 16 Mio

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.

10.89. Supprimer les symboles de débogage

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 *If You Are Going to Chroot* quittez le `chroot` :

```
logout
```

Puis, retournez-y avec :

```
chroot ${CLFS} /tools/bin/env -i \
    HOME=/root TERM=${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,lib64,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.

Si l'espace disque devient très restreint, l'option `--strip-all` peut être utilisée sur les binaires compris dans `{,usr/}{bin,sbin}` pour gagner quelques mégaoctets de plus. N'utilisez pas cette option sur les bibliothèques —cela les détruirait.

Chapitre 11. Initialiser les scripts de démarrage du système

11.1. Introduction

Ce chapitre montre comment installer et configurer le paquet CLFS-Bootscripts. La plupart de ces scripts fonctionne sans modification mais quelques-uns nécessitent des fichiers de configuration supplémentaires car ils dépendent des informations dépendant du matériel.

Les scripts de démarrage compatibles System-V sont utilisés dans ce livre simplement parce qu'ils sont largement utilisés. Pour d'autres options, une astuce détaillant les scripts compatibles BSD est disponible sur <http://hints.cross-lfs.org/index.php/bSD-Init>. Une recherche de « depinit » sur les listes de diffusion CLFS offrira des choix supplémentaires.

Si vous utilisez un autre style de scripts de démarrage, passez ce chapitre et allez directement sur le Making the CLFS System Bootable.

11.2. Scripts de démarrage pour CLFS 2.0-pre2

Le paquet Bootscripts contient un ensemble de scripts de démarrage pour démarrer/arrêter le système CLFS lors de l'amorçage ou de l'arrêt.

11.2.1. Installation des scripts de démarrage

Installez le paquet :

```
make install-bootscripts
```

Vous devrez lancer la commande suivante pour installer le support du réseau :

```
make install-network
```

11.2.2. Contenu des scripts de démarrage

Scripts installés: checkfs, cleanfs, console, functions, halt, ifdown, ifup, localnet, mountfs, mountkernfs, network, rc, reboot, sendsignals, setclock, static, swap, sysklogd, template et udev.

Descriptions courtes

checkfs	Vérifie l'intégrité des systèmes de fichiers avant de les monter (à l'exception des systèmes de fichiers journalisés ou réseau)
cleanfs	Supprime les fichiers qui ne devraient pas être conservés après un redémarrage, tels que ceux compris dans <code>/var/run/</code> et <code>/var/lock/</code> ; il re-crée <code>/var/run/utmp</code> et supprime les fichiers <code>/etc/nologin</code> , <code>/fastboot</code> et <code>/forcefsck</code>
console	Charge la bonne table de correspondance du clavier ; il initialise aussi la police de l'écran
functions	Contient des fonctions communes, telles que la vérification d'erreurs et de statuts, utilisées par les différents scripts de démarrage
halt	Arrête le système
ifdown	Assiste le script network pour l'arrêt des périphériques réseaux
ifup	Assiste le script network pour le démarrage des périphériques réseaux
localnet	Configure le nom d'hôte du système et le périphérique de boucle locale
mountfs	Monte tous les systèmes de fichiers, sauf ceux marqués <i>noauto</i> ou les systèmes réseaux
mountkernfs	Monte les systèmes de fichiers virtuels fournies par le noyau, tels que <code>proc</code>
network	Configure les interfaces réseaux, telles que les cartes réseaux, et configure la passerelle par défaut (lorsque c'est applicable)
rc	Script de contrôle du niveau d'exécution maître. Il est responsable du lancement des autres scripts un par un dans 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 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 (Internet Protocol) statique vers une interface réseau

swap	Active et désactive les fichiers et les partitions d'échange
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	Démarre et arrête le démon udev.

11.3. 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*. Cela peut être bien différent d'un système à un autre, du coup, il ne peut pas être supposé que, parce que cela fonctionne dans une distribution Linux particulière, cela fonctionnera de la même façon dans CLFS. CLFS 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 pour 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 simple 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
```

La commande utilisée pour modifier le niveau d'exécution est **init [niveau_execution]**, où *[niveau_execution]* 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 d'arrêt et de lancement 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 stoppé.

Voici des 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 de nouveau lancé.

reload

La configuration du service est mise à jour. Ceci est utilisé après que le fichier de configuration d'un service a été modifié, 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.

11.4. Configurer le script **setclock**

Le script **setclock** lit le temps sur l'horloge matérielle, aussi connu sous le nom d'horloge BIOS ou CMOS (Complementary Metal Oxide Semiconductor). Si l'horloge matérielle est configurée en UTC, le 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.

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"
# Begin /etc/sysconfig/clock

UTC=1

# End /etc/sysconfig/clock
EOF
```

Une bonne astuce expliquant comment gérer l'horloge sur LFS est disponible sur <http://hints.cross-lfs.org/index.php/time.txt>. Il explique certains concepts comme les fuseaux horaires, UTC et la variable d'environnement TZ.

11.5. Configurer la console Linux

Cette section discute de la configuration des scripts de démarrage **i18n**, initialisant le plan de codage du clavier et la police de la console. Si des caractères non ASCII (par exemple, la livre anglaise et le caractère Euro) ne seront pas utilisés et que le clavier est un clavier US, passez cette section. Sans le fichier de configuration, le script de démarrage **console** ne fera rien.

Le script **i18n** lit le fichier `/etc/sysconfig/i18n` pour des informations de configuration. Décidez 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>). Un fichier `/etc/sysconfig/i18n` préfabriqué avec des paramètres connus pour plusieurs pays a été installé avec le paquet CLFS-Bootscripts, donc vous pouvez décommenter la section appropriée si votre pays est supporté. Si vous avez toujours des doutes, jetez un œil dans le répertoire `/usr/share/consolefonts` pour des polices d'écran valides et `/usr/share/keymaps` pour des plans de codage valides.

Le fichier `/etc/sysconfig/i18n` contient des informations supplémentaires pour vous aider à la configuration..

11.6. Gestion des périphériques et modules sur un système CLFS

Dans *Installing Basic System Software*, nous avons installé le paquet Udev. Avant d'aller dans les détails concernant son fonctionnement, un bref historique des méthodes précédentes de gestion des périphériques est nécessaire.

Les systèmes Linux en général utilisent traditionnellement une méthode de création de périphériques statiques avec laquelle un grand nombre de nœuds périphériques est créé sous `/dev` (quelque fois des milliers de nœuds), que le matériel correspondant existe ou pas. Ceci se 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 périphériques détectés par le noyau obtiennent des nœuds périphériques créés pour eux. Comme ces nœuds périphériques seront créés à chaque lancement du système, ils seront stockés dans un `tmpfs` (un système de fichiers qui réside entièrement en mémoire). Les nœuds périphériques ne requièrent pas beaucoup d'espace disque, donc la mémoire utilisée est négligeable.

11.6.1. 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 le source du noyau, cette méthode de création dynamique de périphérique n'a jamais reçu un support inconditionnel des développeurs du noyau.

Le principal problème de l'approche adoptée par `devfs` était la façon dont il gérât 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 pas imposée par un ou des développeur(s) en particulier. Le système de fichiers `devfs` souffre aussi de conditions particulières inhérentes à son concept et ne peut pas être corrigé sans une revue importante du noyau. Il a aussi été marqué comme obsolète à cause d'un manque de maintenance.

Avec le développement du noyau instable 2.5, sorti ensuite en tant que 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 de l'espace utilisateur, la possibilité de voir un remplacement de l'espace utilisateur pour `devfs` est devenu beaucoup plus réaliste.

11.6.2. Implémentation d'Udev

11.6.2.1. Sysfs

Le système de fichier `sysfs`. 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 leur objet avec `sysfs` 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 qu'à **udev** pour la création des nœuds périphériques.

11.6.2.2. Scripts de démarrage d'Udev

Le script de démarrage **S10udev** s'occupe de créer les nœuds périphériques au lancement de Linux. Le script supprime la gestion des uevents de **/sbin/hotplug** par défaut. On fait cela car le noyau n'a plus besoin de faire appel à un binaire externe. À la place, **udev** écouterait sur une socket netlink les uevents que le noyau fait apparaître. Puis, le script de démarrage copie les nœuds des périphériques statiques qui existent dans **/lib/udev/devices** vers **/dev**. Cela est nécessaire car certains périphériques, répertoires et liens symboliques sont requis avant que les processus de gestion du périphérique dynamique ne soient disponibles pendant les premières étapes du démarrage d'un système. La création des nœuds statiques dans **/lib/udev/devices** fournit aussi un environnement de travail facile pour les périphériques qui ne sont pas supportés par l'infrastructure de gestion des périphériques en dynamique. Ensuite le script de démarrage lance le démon Udev, **udev**, qui agira sur tous les uevents qu'il reçoit. Enfin, le script de démarrage oblige le noyau à répéter des uevents pour chaque périphérique qui a été déjà enregistré puis attend que **udev** les gère.

11.6.2.3. Création de nœuds de périphérique

Pour obtenir le bon nombre majeur ou mineur d'un périphérique, Udev s'appuie sur les informations fournies par **sysfs** dans **/sys**. Par exemple, **/sys/class/tty/vcs/dev** contient la chaîne « 7:0 ». Cette chaîne est utilisée par **udev** 7 et un nombre mineur 0. Les noms et les droits des nœuds sous le répertoire **/dev** sont déterminés par des règles spécifiées dans des fichiers à l'intérieur du répertoire **/etc/udev/rules.d/**. Celles-ci sont numérotées d'une façon similaire au paquet CLFS-Bootscripts. Si **udev** ne peut trouver une règle pour le périphérique qu'il est en train de créer, il attribuera par défaut des droits 660 et la propriété à **root:root**. La documentation sur la syntaxe des fichiers de configuration des règles Udev est disponible dans **/usr/share/doc/udev/writing_udev_rules/index.html**

11.6.2.4. Chargement d'un module

Il se peut que les pilotes des périphériques compilés en module aient des aliases compilés en eux. Les aliases sont visibles dans la sortie du programme **modinfo** et sont souvent liés aux identifiants spécifiques au 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, et il a un alias qui est « pci:v00001319d00000801sv*sd*bc04sc01i* ». Pour la plupart des périphériques, le pilote du bus définit l'alias du pilote qui gèrerait 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 par défaut fournies avec Udev que **udev** 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 *forte* obsolète (et non désiré) 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.

11.6.2.5. Gestion des périphériques dynamiques/montables à chaud

Quand vous connectez un périphérique, comme un lecteur MP3 USB (*Universal Serial Bus*), le noyau reconnaît que le périphérique est maintenant connecté et génère un uevent. Cet uevent est alors géré par **udev** comme décrit ci-dessus.

11.6.3. Problèmes avec le chargement des modules et la création des périphériques

Il existe quelques problèmes connus pour la création automatique des nœuds périphériques :

11.6.3.1. Un module du noyau n'est pas chargé automatiquement

Udev ne chargera un module que s'il a un alias spécifique au bus et si le pilote du bus envoie correctement les alias nécessaires vers `sysfs`. Sinon, il faut organiser le chargement de modules par d'autres moyens. Avec Linux-3.4.17, Udev est connu pour charger les pilotes correctement écrits pour les périphériques INPUT, IDE, PCI, USB, SCSI, SERIO et FireWire.

Pour déterminer si le pilote du périphérique dont vous avez besoin a le support nécessaire pour Udev, lancez **modinfo** avec le nom du module comme argument. Puis, essayez de localiser le répertoire du périphérique sous `/sys/bus` et vérifiez s'il y a un fichier `modalias` là-bas.

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-3.4.17, c'est le cas pour les bus ISA. Attendez que ce problème soit réparé dans les versions ultérieures du noyau.

Udev n'a pas du tout pour but de charger des pilotes « wrappers » (qui emballent un autre pilote) comme *snd-pcm-oss* et des pilotes non matériels comme *loop*.

11.6.3.2. 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 la commande **modprobe** pour charger le wrapper après qu'Udev ait chargé le module emballé. Pour cela, ajoutez une ligne « install » dans `/etc/modprobe.conf`. Par exemple :

```
install snd-pcm /sbin/modprobe -i snd-pcm ; \
    /sbin/modprobe snd-pcm-oss ; true
```

Si le module en question n'est pas un emballage et s'avère utile en tant que tel, configurez le script de démarrage **S05modules** 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. Cela fonctionne aussi pour les modules emballage, mais ce n'est pas optimal dans ce cas.

11.6.3.3. Udev charge un module indésirable

Ne compilez pas le module, ou mettez-le en liste noire dans le fichier `/etc/modprobe.conf` comme cela est 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**.

11.6.3.4. 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 écrite avec des lacunes peut correspondre à un disque SCSI (comme désiré) et au périphérique générique SCSI correspondant (de façon incorrecte) du fabricant. Trouvez la règle défectueuse et rendez-la plus précise à l'aide de **udevadm info**.

11.6.3.5. 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 la mettant dans le fichier `/etc/udev/rules.d/10-wait_for_sysfs.rules`. Merci d'informer la liste de développement de CLFS si vous faites ainsi et que cela vous aide.

11.6.3.6. Udev ne crée pas de périphérique

Le texte ci-après assume 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 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 de tierces parties 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 `S10udev`.

11.6.3.7. Le nommage des périphériques change de manière aléatoire après le redémarrage

Cela est dû au fait que 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 espérer 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 Section 11.7, « Création de liens symboliques personnalisés vers les périphériques » et Networking Configuration pour des exemples.

11.6.4. Lecture utile

Des documentations supplémentaires sont disponibles sur les sites suivants :

- The `sysfs` Filesystem <http://www.kernel.org/pub/linux/kernel/people/mochel/doc/papers/ols-2005/mochel.pdf> (NdT : Le système de fichier `sysfs`)

11.7. Création de liens symboliques personnalisés vers les périphériques

11.7.1. Liens symboliques pour le CD-ROM

Certains logiciels que vous pourriez vouloir installer plus tard (comme des 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`. Pour chacun de vos périphériques 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`.

Il y a deux approches pour créer des liens symboliques. La première consiste à utiliser le nom de modèle et le numéro de série, la seconde est basée sur l'emplacement du périphérique sur le bus. Si vous allez utiliser la première approche, créez un fichier qui ressemble à ce qui suit :

```
cat >/etc/udev/rules.d/82-cdrom.rules << EOF

# Liens symboliques pour le lecteur CD-ROM personnalisés
SUBSYSTEM=="block", ENV{ID_MODEL}=="SAMSUNG_CD-ROM_SC-148F", \
    ENV{ID_REVISION}=="PS05", SYMLINK+="cdrom"
SUBSYSTEM=="block", ENV{ID_MODEL}=="PHILIPS_CDD5301", \
    ENV{ID_SERIAL}=="5VO1306DM00190", SYMLINK+="cdrom1 dvd"

EOF
```



Remarque

Bien que les exemples de ce livre fonctionnent correctement, gardez à l'esprit qu'Udev ne reconnaît pas les antislash pour poursuivre une ligne. Si vous modifiez des règles Udev avec un éditeur, prenez garde à laisser chaque règle sur une ligne physique.

De cette façon, les liens symboliques resteront bons même si vous déplacez les périphériques dans des positions différentes sur le bus IDE mais le lien symbolique `/dev/cdrom` ne sera pas créé si vous remplacez le vieux SAMSUNG CD-ROM par un nouveau lecteur.

La clé `SUBSYSTEM=="block"` est nécessaire afin d'éviter une correspondance entre les périphériques génériques SCSI. Sans cela, avec des lecteurs de CD-ROM SCSI, les liens symboliques pointeront tantôt vers les bons périphériques `/dev/srX`, tantôt vers `/dev/sgX`, ce qui est faux.

La seconde approche donne :

```
cat >/etc/udev/rules.d/82-cdrom.rules << EOF

# Liens symboliques pour le lecteur CD-ROM personnalisés
SUBSYSTEM=="block", ENV{ID_TYPE}=="cd", \
    ENV{ID_PATH}=="pci-0000:00:07.1-ide-0:1", SYMLINK+="cdrom"
SUBSYSTEM=="block", ENV{ID_TYPE}=="cd", \
    ENV{ID_PATH}=="pci-0000:00:07.1-ide-1:1", SYMLINK+="cdrom1 dvd"

EOF
```

De cette façon, les liens symboliques demeureront corrects même si vous remplacez des lecteurs par des modèles différents mais que vous placez sur les anciennes positions sur le bus IDE. La clé `ENV{ID_TYPE}=="cd"` s'assure que le lien symbolique disparaisse si vous mettez quelque chose d'autre qu'un lecteur de CD-ROM dans une telle position sur le bus.

Bien entendu, il est possible de mélanger les deux approches.

11.7.2. Gestion des périphériques dupliqués

Comme expliqué dans la Section 11.6, « Gestion des périphériques et modules sur un système CLFS », l'ordre dans lequel les périphériques ayant la même fonction apparaissent dans `/dev` est essentiellement aléatoire. Par exemple si vous avez une webcam en USB et un tuner 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 être corrigé en créant des règles udev pour des liens symboliques constants personnalisés. Le cas des cartes réseau est traité séparément dans la Networking Configuration et vous pouvez trouver la configuration des cartes son dans *CBLFS*.

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 constants pour webcam et tuner
KERNEL=="video*", SYSFS{idProduct}=="1910", SYSFS{idVendor}=="0d81", \
    SYMLINK+="webcam"
KERNEL=="video*", SYSFS{device}=="0x036f", SYSFS{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.

Vous pouvez trouver plus d'informations sur l'écriture de règles Udev dans `/usr/share/doc/udev/writing_udev_rules/index.html`.

11.8. 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 paramètres globaux.

Un shell interactif de connexion est lancé après une connexion réussie, en utilisant `/bin/login`, par la lecture du fichier `/etc/passwd`. Un shell interactif sans connexion est lancé en ligne de commande (c'est-à-dire `[prompt]$ /bin/bash`). Un shell non interactif est habituellement présent quand un script shell est en cours d'exécution. Il est non interactif parce qu'il traite un script et n'attend pas une saisie de l'utilisateur entre les commandes.

Pour plus d'informations, voir **info bash** sous la section *Bash Startup Files and Interactive Shells* (Fichiers de démarrage de Bash et shells interactifs), et *Bash Startup Files* dans *CBLFS*.

Les fichiers `/etc/profile` et `~/.bash_profile` sont lus quand le shell est appelé en tant que shell interactif de connexion. Dans la section suivante, un `/etc/profile` sera créé pour paramétrer les informations de locale.

11.9. Setting Up Locale Information

The base `/etc/profile` below sets some environment variables necessary for native language support. Setting them properly results in:

- The output of programs translated into the native language
- Correct classification of characters into letters, digits and other classes. This is necessary for **bash** to properly accept non-ASCII characters in command lines in non-English locales
- The correct alphabetical sorting order for the country
- Appropriate default paper size
- Correct formatting of monetary, time, and date values

This script also sets the `INPUTRC` environment variable that makes Bash and Readline use the `/etc/inputrc` file created earlier.

Replace `[LL]` below with the two-letter code for the desired language (e.g., « en ») and `[CC]` with the two-letter code for the appropriate country (e.g., « GB »). `[charmap]` should be replaced with the canonical charmap for your chosen locale.

The list of all locales supported by Glibc can be obtained by running the following command:

```
locale -a
```

Locales can have a number of synonyms, e.g. « ISO-8859-1 » is also referred to as « iso8859-1 » and « iso88591 ». Some applications cannot handle the various synonyms correctly, so it is safest to choose the canonical name for a particular locale. To determine the canonical name, run the following command, where `[locale name]` is the output given by **locale -a** for your preferred locale (« en_US.utf8 » in our example).

```
LC_ALL=[locale name] locale charmap
```

For the « en_US.utf8 » locale, the above command will print:

```
UTF-8
```

This results in a final locale setting of « en_US.UTF-8 ». It is important that the locale found using the heuristic above is tested prior to it being added to the Bash startup files:

```
LC_ALL=[locale name] locale territory
LC_ALL=[locale name] locale language
LC_ALL=[locale name] locale charmap
LC_ALL=[locale name] locale int_curr_symbol
LC_ALL=[locale name] locale int_prefix
```

The above commands should print the language name, the character encoding used by the locale, the local currency, and the prefix to dial before the telephone number in order to get into the country. If any of the commands above fail with a message similar to the one shown below, this means that your locale was either not installed in Chapter 10 or is not supported by the default installation of Glibc.

```
locale: Cannot set LC_* to default locale: No such file or directory
```

If this happens, you should either install the desired locale using the **localedef** command, or consider choosing a different locale. Further instructions assume that there are no such error messages from Glibc.

Some packages beyond CLFS may also lack support for your chosen locale. One example is the X library (part of the X Window System), which outputs the following error message:

```
Warning: locale not supported by Xlib, locale set to C
```

Sometimes it is possible to fix this by removing the charmap part of the locale specification, as long as that does not change the character map that Glibc associates with the locale (this can be checked by running the **locale charmap** command in both locales). For example, one would have to change "de_DE.ISO-8859-15@euro" to "de_DE@euro" in order to get this locale recognized by Xlib.

Other packages can also function incorrectly (but may not necessarily display any error messages) if the locale name does not meet their expectations. In those cases, investigating how other Linux distributions support your locale might provide some useful information.

Once the proper locale settings have been determined, create the `/etc/profile` file:

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

export LANG=[ll]_[CC].[charmap]
export INPUTRC=/etc/inputrc

# End /etc/profile
EOF
```

Setting the keyboard layout, screen font, and locale-related environment variables are the only internationalization steps needed to support locales that use ordinary single-byte encodings and left-to-right writing direction. UTF-8 has been tested on the English, French, German, Italian, and Spanish locales. All other locales are untested. If you discover issues with any other locale please open a ticket in our Trac system.

Some locales need additional programs and support. CLFS will not be supporting these locales in the book. We welcome the support for these other locales via <http://cblfs.cross-lfs.org/>.

11.10. 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** sous 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>

# Ne pas tout écrire sur une seule ligne
set horizontal-scroll-mode Off

# Activer les entrées 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 (aucun), visible ou audible
set bell-style none

# Toutes les indications qui suivent font correspondre la séquence
# d'échappement contenue dans le 1er argument à la fonction
# spécifique de readline

"\eOd": backward-word
"\eOc": forward-word

# Pour la console linux
"\e[1~": beginning-of-line
"\e[4~": end-of-line
"\e[5~": beginning-of-history
"\e[6~": end-of-history
"\e[3~": delete-char
"\e[2~": quoted-insert

# Pour xterm
"\eOH": beginning-of-line
"\eOF": end-of-line

# Pour Konsole
"\e[H": beginning-of-line
"\e[F": end-of-line

# Fin de /etc/inputrc
EOF
```

Chapitre 12. Configuration du réseau

12.1. Configurer le script localnet

Une partie du boulot 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 entrez le nom du système en lançant :

```
echo "HOSTNAME=[c1fs]" > /etc/sysconfig/network
```

Vous devez remplacer `[c1fs]` par le nom donné à l'ordinateur. N'entrez pas le nom de domaine pleinement qualifié (FQDN) ici. On mettra cette information dans le fichier `/etc/hosts` dans la section suivante.

12.2. Personnaliser le fichier `/etc/hosts`

Si une carte réseau doit être configurée, choisissez l'adresse IP, le nom de domaine pleinement qualifié 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'est-à-dire que vous avez enregistré un domaine et un bloc valide d'adresses IP qui vous est affecté—la plupart des utilisateurs n'ont pas ceci), vous devez vous assurer que l'adresse IP se trouve dans la plage d'adresses réservée aux réseaux privés. Les plages valides sont :

Class	Networks
A	10.0.0.0
B	172.16.0.0 through 172.31.0.255
C	192.168.0.0 through 192.168.255.255

Une adresse IP valide pourrait être 192.168.1.1. Un nom de domaine pleinement qualifié pour cette adresse IP pourrait être `www.linuxfromscratch.org` (non recommandé car c'est une adresse de domaine enregistrée et cela pourrait entraîner des problèmes de serveur de nom de domaine).

Même si vous ne possédez pas de carte réseau, un nom de domaine pleinement qualifié est toujours requis. Certains programmes en ont besoin pour fonctionner correctement.

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

```
cat > /etc/hosts << "EOF"
# Début de /etc/hosts (network card version)

127.0.0.1 localhost
[192.168.1.1] [<HOSTNAME>.example.org] [HOSTNAME]

# Fin de /etc/hosts (network card version)
EOF
```

Les valeurs `[192.168.1.1]` et `[<nom d'hôte>.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).

Si vous n'avez pas de carte réseau, créez le fichier `/etc/hosts` en lançant :

```
cat > /etc/hosts << "EOF"
# Début de /etc/hosts (no network card version)

127.0.0.1 [<HOSTNAME>.example.org] [HOSTNAME] localhost

# Fin de /etc/hosts (no network card version)
EOF
```

12.3. Création du fichier `/etc/resolv.conf`

12.3.1. Création du fichier `/etc/resolv.conf`

Si vous allez connecter le système à Internet, vous aurez besoin d'un minimum de résolution de nom de service de nom de domaine, *Domain Name Service* (DNS) pour résoudre des noms de domaine Internet selon les adresses IP et vis versa. La meilleure façon de faire cela est de placer l'adresse IP du serveur de DNS, disponible auprès du FAI ou de l'administrateur du réseau, dans le fichier `/etc/resolv.conf`. Si au moins une de vos interfaces réseau va être configurée par DHCP, il se peut que vous n'ayez pas besoin de créer ce fichier. DHCPD écrasera ce fichier par défaut lorsqu'il obtiendra une nouvelle adresse attribuée par le serveur DHCP. Si vous souhaitez configurer manuellement vos interfaces réseau ou si vous paramétrez manuellement votre DNS en utilisant DHCP, 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 de votre nom de serveur primaire]
nameserver [adresse IP de votre nom de serveur secondaire]

# Fin de /etc/resolv.conf
EOF
```

Remplacez *[adresse IP du nom de serveur]* par l'adresse IP du DNS la plus adaptée à la configuration. Souvent il y aura plus d'une entrée (sont requis des serveurs secondaires pour la fonctionnalité fallback). Si vous n'avez besoin ou ne voulez qu'un serveur DNS, supprimez la seconde ligne *nameserver* du fichier. L'adresse IP peut être aussi un routeur sur le réseau local.

12.4. Réseau DHCP ou Statique ?

Cette section ne vaut que si une carte réseau va être configurée. Si vous n'avez pas besoin de configurer une interface réseau, vous pouvez passer à Making the CLFS System Bootable.

Vous pouvez configurer votre réseau par deux manières différentes. Le dynamique vous permettra de tirer parti d'un serveur DHCP pour obtenir toutes vos informations de configuration. En statique, vous devenez responsable du paramétrage de vos options.

Pour configurer une interface statique, suivez Section 12.5, « Configuration d'un réseau statique ».

Pour configurer une interface DHCP, suivez Section 12.6, « DHCPD-5.5.6 ».

12.5. Configuration d'un réseau statique

12.5.1. Création des fichiers de configuration de l'interface réseau statique

Les interfaces qui sont activées et désactivées par le script `network` dépend des fichiers et des répertoires dans la hiérarchie `/etc/sysconfig/network-devices`. Ce répertoire devrait contenir un sous-répertoire par interface à configurer, comme `ifconfig.xyz`, où « xyz » est un nom d'interface réseau. À l'intérieur de ce répertoire, il y aurait des fichiers qui définissent les attributs de cette interface, comme son/ses adresse(s) IP, ses masques de sous-réseau, et autres.

La commande suivante crée un modèle de fichier `ipv4` pour le périphérique `eth0` :

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

Vous devez modifier les valeurs de ces variables dans chaque fichier pour correspondre à la bonne configuration. Si la variable `ONBOOT` est réglée sur « yes », le script `network` activera la carte interface réseau, *Network Interface Card* (NIC) pendant le démarrage du système. S'il est réglé sur autre chose que « yes » la NIC sera ignorée par le script `network` et non activée.

La variable `SERVICE` définit la méthode utilisée pour obtenir l'adresse IP. Le paquet `CLFS-Bootscripts` a un format d'allocation d'IP modulaire et la création de fichiers supplémentaires dans le répertoire `/etc/sysconfig/network-devices/services` permet d'autres méthodes d'allocation.

La variable `GATEWAY` devrait contenir la passerelle par défaut pour l'adresse IP, si elle existe. Sinon, commentez toute la variable.

La variable `PREFIX` doit contenir le nombre de bits utilisés dans le sous-réseau. Chaque octet dans une adresse IP fait 8 bits. Si le masque de réseau du sous-réseau est `255.255.255.0`, il utilise les trois premiers octets (24 bits) pour spécifier le numéro réseau. Si le masque de réseau est `255.255.255.240`, il utiliserait les 28 premiers bits. Les préfixes plus longs que 24 bits sont souvent utilisés dans les fournisseurs d'accès Internet (FAI) ADSL et câblé. Dans cet exempl (PREFIX=24), le masque réseau est `255.255.255.0`. Ajustez la variable `PREFIX` selon votre sous-réseau spécifique.

Pour configurer une autre interface DHCP, suivez Section 12.7, « Configuration d'un réseau DHCP ».

12.6. DHCPD-5.5.6

Le paquet DHCPD fournit un client DHCP pour la configuration réseau.

12.6.1. Installation de DHCPD



Remarque

Cette construction est facultative. Vous n'en avez besoin que si comptez utiliser un serveur DHCP pour configurer automatiquement au moins une interface réseau sur votre système.

Préparez la compilation de DHCPD :

```
CC="gcc ${BUILD64}" ./configure --prefix=/usr --sbindir=/sbin \
  --sysconfdir=/etc --dbdir=/var/lib/dhcpd --libexecdir=/usr/lib/dhcpd
```

Compilez le paquet :

```
make
```

Ce paquet est fourni sans suite de tests.

Installez le paquet :

```
make install
```

12.6.2. Contenu de dhcpd

Fichiers installés: dhcpd

Descriptions courtes

dhcpd dhcpd est une adaptation du client DHCP tel que spécifié dans la RFC 2131. Il obtient les informations de l'hôte depuis un serveur DHCP et il configure automatiquement l'interface réseau.

12.7. Configuration d'un réseau DHCP

12.7.1. Créez les fichiers de configuration de l'interface réseau DHCP

D'abord, installez le service à partir du paquet CLFS Bootscripts :

```
tar -xvf bootscripts-cross-lfs-2.0-pre2.tar.bz2
cd bootscripts-cross-lfs-2.0-pre2
make install-service-dhcpd
```

Enfin, créez le fichier de configuration `/etc/sysconfig/network-devices/ifconfig.eth0/dhcpd` en utilisant les commandes suivantes. Ajustez selon vos besoins en fonction des interfaces supplémentaires :

```
cd /etc/sysconfig/network-devices &&
mkdir -v ifconfig.eth0 &&
cat > ifconfig.eth0/dhcpd << "EOF"
ONBOOT="yes"
SERVICE="dhcpd"

# Commande de démarrage pour for DHCPD
DHCP_START="-q"

# Commande d'arrêt pour DHCPD
DHCP_STOP="-k"
EOF
```

Vous devez modifier les valeurs de ces variables dans chaque fichier pour correspondre à la bonne configuration. Si la variable `ONBOOT` est réglée sur « yes », le script `network` activera la carte interface réseau, *Network Interface Card* (NIC) pendant le démarrage du système. S'il est réglé sur autre chose que « yes » la NIC sera ignorée par le script `network` et non activée.

La variable `SERVICE` définit la méthode utilisée pour obtenir l'adresse IP. Le paquet CLFS-Bootscripts a un format d'allocation d'IP modulaire et la création de fichiers supplémentaires dans le répertoire `/etc/sysconfig/network-devices/services` permet d'autres méthodes d'allocation.

Les arguments de variables `DHCP_START` et `DHCP_STOP` sont passés à `dhcpd` lors du lancement et de l'arrêt du service. Vous pouvez trouver plus d'informations sur ce que vous pouvez passer dans la page de man de `dhcpd(8)`.

Pour configurer une autre Interface Statique, suivez Section 12.5, « Configuration d'un réseau statique ».

Chapitre 13. Rendre le système CLFS amorçable

13.1. Introduction

Il est temps de rendre amorçable le système CLFS. Ce chapitre traite de la création d'un fichier `fstab`, de la construction d'un noyau pour le nouveau système CLFS et de l'installation du chargeur de démarrage afin que le système CLFS puisse être sélectionné au démarrage.

13.2. Créer le fichier `/etc/fstab`

Le fichier `/etc/fstab` est utilisé par quelques programmes pour déterminer les partitions à monter par défaut, dans quel ordre, et quels systèmes de fichiers sont à vérifier (pour des erreurs d'intégrité). Créez une nouvelle table des systèmes de fichiers comme ceci :

```
cat > /etc/fstab << "EOF"
# Début de /etc/fstab

# Sys. de fich.  point montage  type  options                montage  ordre
#                fsck

/dev/[xxx]      /                [fff]  defaults                1        1
/dev/[yyy]      swap            swap    pri=1                    0        0
proc            /proc           proc    defaults                0        0
sysfs           /sys            sysfs   defaults                0        0
devpts          /dev/pts        devpts  gid=4,mode=620          0        0
shm             /dev/shm        tmpfs   defaults                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 adaptées à votre système, par exemple `hda2`, `hda5` et `ext2`. Pour des détails sur les six champs de ce fichier, voir **man 5 fstab**.

Le point de montage `/dev/shm` pour `tmpfs` est inclu pour permettre l'activation de la mémoire partagée POSIX. Le noyau doit disposer du support requis en interne pour fonctionner (plus d'informations là-dessus dans la prochaine section). Merci de noter qu'actuellement très peu de logiciels utilise la mémoire partagée POSIX. Donc, vous pouvez considérer le point de montage `/dev/shm` comme optionnel. Pour plus d'informations, voir `Documentation/filesystems/tmpfs.txt` dans le répertoire des sources du noyau.

13.3. Linux-3.4.17

Le paquet Linux contient le noyau Linux.

13.3.1. Installation du noyau

La construction du noyau implique quelques étapes — la configuration, la compilation et l'installation. Lisez le fichier README dans l'arborescence des sources du noyau pour des méthodes alternatives de à celle utilisée par le livre pour configurer le noyau.

Préparez la compilation en lançant la commande suivante :

```
make mrproper
```

Ceci garantit que l'arborescence du noyau est absolument propre. L'équipe du noyau recommande que cette commande soit exécutée avant chaque compilation du noyau. Ne pensez pas que l'arborescence des sources est propre après la décompression.

Configurez le noyau avec une interface en menus. Merci de noter que le script de démarrage d'Udev exige que "rtc", "tmpfs" et "devtmpfs" soient activés et construits en dur dans le noyau et non en modules. CBLFS contient des informations sur les exigences de configuration particulières des paquets hors CLFS, sur <http://cblfs.cross-lfs.org/>:

```
make menuconfig
```

Sinon, **make oldconfig** pourrait être plus adapté à certaines situations. Voir le fichier README pour plus d'informations.

Si vous le souhaitez, passez la configuration du noyau en copiant le fichier de configuration du noyau, `.config`, du système hôte (s'il est disponible) dans le répertoire racine des sources déballées du noyau. Toutefois, nous ne vous recommandons pas cette option. Il vaut souvent mieux explorer tous les menus de configuration et créer de zéro la configuration du noyau.

Compilez l'image et les modules du noyau :

```
make
```

If using kernel modules, an `/etc/modprobe.conf` file may be needed. Information pertaining to modules and kernel configuration is located in the kernel documentation in the `Documentation` directory of the kernel sources tree. Also, `modprobe.conf(5)` may be of interest.

Be very careful when reading other documentation relating to kernel modules because it usually applies to 2.4.x kernels only. As far as we know, kernel configuration issues specific to Hotplug and Udev are not documented. The problem is that Udev will create a device node only if Hotplug or a user-written script inserts the corresponding module into the kernel, and not all modules are detectable by Hotplug. Note that statements like the one below in the `/etc/modprobe.conf` file do not work with Udev:

```
alias char-major-XXX some-module
```

Because of the complications with Udev and modules, we strongly recommend starting with a completely non-modular kernel configuration, especially if this is the first time using Udev.

Install the modules, if the kernel configuration uses them:

```
make modules_install
```

Installez le firmware si la configuration du noyau en utilise un :

```
make firmware_install
```

After kernel compilation is complete, additional steps are required to complete the installation. Some files need to be copied to the `/boot` directory.

Issue the following command to install the kernel:

```
cp -v arch/x86_64/boot/bzImage /boot/vmlinuz-clfs-3.4.17
```

`System.map` is a symbol file for the kernel. It maps the function entry points of every function in the kernel API, as well as the addresses of the kernel data structures for the running kernel. Issue the following command to install the map file:

```
cp -v System.map /boot/System.map-3.4.17
```

The kernel configuration file `.config` produced by the **make menuconfig** step above contains all the configuration selections for the kernel that was just compiled. It is a good idea to keep this file for future reference:

```
cp -v .config /boot/config-3.4.17
```

It is important to note that the files in the kernel source directory are not owned by `root`. Whenever a package is unpacked as user `root` (like we do inside the final-system build environment), the files have the user and group IDs of whatever they were on the packager's computer. This is usually not a problem for any other package to be installed because the source tree is removed after the installation. However, the Linux source tree is often retained for a long time. Because of this, there is a chance that whatever user ID the packager used will be assigned to somebody on the machine. That person would then have write access to the kernel source.

If the kernel source tree is going to be retained, run **chown -R 0:0** on the `linux-3.4.17` directory to ensure all files are owned by user `root`.



Avertissement

Some kernel documentation recommends creating a symlink from `/usr/src/linux` pointing to the kernel source directory. This is specific to kernels prior to the 2.6 series and *must not* be created on a CLFS system as it can cause problems for packages you may wish to build once your base CLFS system is complete.

Also, the headers in the system's `include` directory should *always* be the ones against which Glibc was compiled and should *never* be replaced by headers from a different kernel version.

13.3.2. Contents of Linux

Fichiers installés:	<code>config-[linux-version]</code> , <code>clfskernel-[linux-version]</code> , et <code>System.map-[linux-version]</code>
Répertoire installé:	<code>/lib/modules</code>

Short Descriptions

<code>config-[linux-version]</code>	Contains all the configuration selections for the kernel
<code>clfskernel-[linux-version]</code>	The engine of the Linux system. When turning on the computer, the kernel is the first part of the operating system that gets loaded. It detects and initializes all components of the computer's hardware, then makes these components available as a tree of files to the software and turns a single CPU into a multitasking machine capable of running scores of programs seemingly at the same time.
<code>System.map-[linux-version]</code>	A list of addresses and symbols; it maps the entry points and addresses of all the functions and data structures in the kernel

13.4. Rendre le système CLFS amorçable

Votre système CLFS flambant neuf est presque terminé. Une des dernières choses à faire est de vous assurer que le système peut démarrer correctement. Les instructions ci-dessous s'appliquent aux architectures x86 et x86_64, c'est-à-dire la plupart des PCs. Des informations sur le « chargement de l'amorce » pour d'autres architectures devraient être disponibles aux endroits habituels des ressources spécifiques à ces architectures.

Le chargement au démarrage peut être un sujet complexe, donc quelques mots de prudence sont utiles. Familiarisez-vous avec le chargeur de démarrage actuel et tous les autres systèmes d'exploitation présent sur le(s) disque(s) dur(s) et qui doit/doivent être amorcé(s). Assurez-vous d'avoir un disque de démarrage d'urgence disponible pour « secourir » l'ordinateur s'il devient inutilisable (inamorçable).



Avertissement

La commande suivante écrasera le chargeur de démarrage actuel. Ne lancez pas la commande si vous ne désirez pas cela, par exemple si vous utilisez d'un chargeur de démarrage tiers pour gérer le Master Boot Record (MBR). Dans ce cas, il serait plus logique d'installer GRUB dans le « boot sector » de la partition CLFS. Dans ce cas, la commande suivante deviendrait **grub-install /dev/sda2**.

Demandez à GRUB de s'installer dans la MBR de sda :

```
grub-install /dev/sda
```

Ensuite, il faut générer une configuration pour GRUB. Dans les précédentes versions de grub, nous pouvions créer la configuration à la main, mais avec GRUB2, nous pouvons générer `grub.cfg` automatiquement. Vous pouvez faire cela avec la commande suivante :

```
grub-mkconfig -o /boot/grub/grub.cfg
```

Chapitre 14. La fin

14.1. La fin

Bien joué ! Le nouveau système CLFS est installé. Nous vous souhaitons de bien vous amuser avec votre nouveau système Linux compilé et personnalisé rutilant.

Une bonne idée serait de créer un fichier `/etc/clfs-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 GIT-20130401 > /etc/clfs-release
```

14.2. Client de téléchargement

La construction du système final n'installe pas de client FTP ou HTTP pour télécharger des fichiers.

Parmi les clients suggérés, on a :

- Curl <http://cblfs.cross-lfs.org/index.php/Curl>
- Inetutils <http://cblfs.cross-lfs.org/index.php/Inetutils>
- LFTP <http://lftp.yar.ru/>
- Links <http://cblfs.cross-lfs.org/index.php/Links>
- Lynx <http://cblfs.cross-lfs.org/index.php/Lynx>
- NcFTP Client <http://cblfs.cross-lfs.org/index.php/Ncftp>
- Wget <http://cblfs.cross-lfs.org/index.php/Wget>
- BASH - Un utilisateur peut utiliser les redirections net (si elles ne sont pas désactivées au moment de la construction de bash dans le système final) pour télécharger wget ou un autre programme.

```
cat > download.sh << "EOF"
#!/bin/bash

WGET_VERSION='1.14'
WGET_HOSTNAME='ftp.gnu.org'
exec {HTTP_FD}<>/dev/tcp/${WGET_HOSTNAME}/80
echo -ne "GET /gnu/wget/wget-${WGET_VERSION}.tar.xz HTTP/1.1\r\nHost: "\
${WGET_HOSTNAME}'\r\nUser-Agent: '\
'bash/'${BASH_VERSION}'\r\n\r\n' >&${HTTP_FD}
sed -e '1,/^.$/d' <&${HTTP_FD} >wget-${WGET_VERSION}.tar.xz
EOF
```

- GAWK

```
cat > gawkdl.sh << "EOF"
#!/bin/bash

gawk 'BEGIN {
    NetService = "/inet/tcp/0/mirror.anl.gov/80"
    print "GET /pub/gnu/wget/wget-1.14.tar.xz" |& NetService
    while ((NetService |& getline) > 0)
        print $0
    close(NetService)
}' > binary

gawk '{q=p;p=$0}NR>1{print q}END{ORS = ""; print p}' binary > wget-1.14.tar.xz

rm binary
EOF
```

- PERL avec HTTP::Tiny (Inclus dans l'installation de PERL du système final).

```
cat > download.pl << "EOF"
#!/usr/bin/perl

use HTTP::Tiny;
my $http = HTTP::Tiny->new;
my $response;

$response = $http->mirror('http://ftp.gnu.org/gnu/wget/wget-1.14.tar.xz', 'wget-1.14.tar.xz');
die "Failed!\n" unless $response->{success};
print "Unchanged!\n" if $response->{status} eq '304';
EOF
```

Ou utilisez ceci :

```
perl -MHTTP::Tiny -E 'say HTTP::Tiny->new->get(shift)->{content}' "http://ftp.gnu.org/gnu/wget/wget-1.14.tar.xz"
perl -e 'local $/; $_ = <>; s/\n$//; print' binary > wget-1.14.tar.xz
rm binary
```

- PERL avec LWP : lancez **cpan** et configurez à la main le client. Lancez **install LWP** dans le shell CPAN.

Référez-vous à <http://www.bioinfo-user.org.uk/dokuwiki/doku.php/projects/wgetpl> concernant wgetpl.

14.3. Redémarrer le système

Si vous avez construit votre système final en utilisant la méthode du démarrage, lancez simplement **shutdown -r now** pour redémarrer à nouveau en utilisant votre noyau nouvellement construit à la place de celui minimal actuellement utilisé. Si vous vous êtes chrooté, il y a quelques étapes en plus.

Le système que vous avez créé dans ce livre est vraiment minimal et a toutes les chances de ne pas avoir les fonctionnalités dont vous aurez besoin pour continuer. En installant quelques autres paquets à partir de CBLFS 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 CLFS. Installer un navigateur web en mode texte, comme Lynx, vous permettra de voir facilement le site Internet CBLFS dans un terminal virtuel tout en construisant des

paquets dans un autre. Le paquet GPM vous permettra aussi de réaliser des actions de copier/coller dans vos terminaux virtuels. Enfin, 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 Dhcpd ou PPP pourrait aussi être utile.

Maintenant qu'on a dit ça, démarrons notre toute nouvelle installation CLFS pour la première fois ! Tout d'abord, quittez l'environnement chroot :

```
logout
```

Puis, démontez les systèmes de fichiers virtuels :

```
umount -v ${CLFS}/dev/pts
umount -v ${CLFS}/dev/shm
umount -v ${CLFS}/dev
umount -v ${CLFS}/proc
umount -v ${CLFS}/sys
```

Démontez le système de fichiers CLFS :

```
umount -v ${CLFS}
```

Si plusieurs partitions ont été créées, démontez les autres partitions avant de démonter la principale, comme ceci :

```
umount -v ${CLFS}/usr
umount -v ${CLFS}/home
umount -v ${CLFS}
```

Maintenant, redémarrez le système avec :

```
shutdown -r now
```

En supposant que le chargeur de démarrage a été initialisé comme indiqué plus tôt, *CLFS GIT-20130401* va démarrer automatiquement.

Quand le redémarrage est terminé, le système CLFS est prêt à être utilisé et des logiciels peuvent enfin être installés pour satisfaire vos besoins.

14.4. Et maintenant ?

Merci d'avoir lu le livre CLFS. Nous espérons que vous avez trouvé ce livre utile et que vous avez appris plus sur le processus de création d'un système.

Maintenant que le système CLFS est installé, vous êtes peut-être en train de vous demander « Quelle est la suite ? » 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, certains d'entre eux sont indiqués ci-dessous :

- Freecode (<http://freecode.com/>)

Freecode peut vous prévenir (par email) des nouvelles versions de paquetages 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 pour révéler complètement les problèmes de sécurité. Elle publie les problèmes de sécurité qui viennent d'être découverts et, occasionnellement, des corrections potentielles. Les informations de souscription sont disponibles sur <http://www.securityfocus.com/archive>.

- Community Driven Beyond Linux From Scratch

Le wiki Community Driven Beyond Linux From Scratch (au-delà de Linux From Scratch par la communauté) couvre les procédures d'installation d'un grand nombre de logiciels en dehors du livre CLFS. CBLFS est destiné spécifiquement à fonctionner avec le livre CLFS et contient toutes les informations nécessaires pour continuer les constructions de la même manière que ce que pratique CLFS. C'est le projet de la communauté amenant à cela, ce qui signifie que n'importe qui peut contribuer et fournir des mises à jour. Le projet CBLFS se situe sur <http://cblfs.cross-lfs.org/>.

- Astuces CLFS

Les astuces CLFS sont une collection de documents éducatifs soumis par des volontaires à la communauté CLFS. Ces astuces sont disponibles sur <http://hints.cross-lfs.org/index.php/>.

- Listes de diffusion

Il existe plusieurs listes de diffusion CLFS auxquelles vous pouvez vous abonner si vous cherchez de l'aide, voulez rester à jour avec les derniers développements, voulez contribuer au projet et plus. Voir Chapitre 1 - Listes de diffusion pour plus d'informations.

- Le projet de documentation Linux (Linux Documentation Project)

Le but du TLDP est de collaborer à tous les problèmes relatifs à la documentation sur Linux. Le TLDP offre une large collection de guides pratiques, livres et pages man. Il est disponible sur <http://www.tldp.org/>.

Partie VI. Annexes

Annexe A. Acronymes et termes

ABI	Application Binary Interface
ALSA	Advanced Linux Sound Architecture
API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
ATA	Advanced Technology Attachment (see IDE)
BIOS	Basic Input/Output System
bles	manipulate a filesystem so that OF will boot from it
BSD	Berkeley Software Distribution
CBLFS	Community Driven Beyond Linux From Scratch
chroot	change root
CLFS	Cross-Compiled Linux From Scratch
CMOS	Complementary Metal Oxide Semiconductor
COS	Class Of Service
CPU	Central Processing Unit
CRC	Cyclic Redundancy Check
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name Service
EGA	Enhanced Graphics Adapter
ELF	Executable and Linkable Format
EOF	End of File
EQN	equation
ext2	second extended file system
ext3	third extended file system
ext4	fourth extended file system
FAQ	Frequently Asked Questions
FHS	Filesystem Hierarchy Standard
FIFO	First-In, First Out
FQDN	Fully Qualified Domain Name
FTP	File Transfer Protocol
Gio	Giga octet informatique
GCC	GNU Compiler Collection
GID	Group Identifier
GMT	Greenwich Mean Time
HTML	Hypertext Markup Language
IDE	Integrated Drive Electronics
IEEE	Institute of Electrical and Electronic Engineers
IO	Input/Output

IP	Internet Protocol
IPC	Inter-Process Communication
IRC	Internet Relay Chat
ISO	International Organization for Standardization
ISP	Internet Service Provider
KB	Kilo octet informatique
LED	Light Emitting Diode
LFS	Linux From Scratch
LSB	Linux Standard Base
MB	Méga octet informatique
MBR	Master Boot Record
MD5	Message Digest 5
NIC	Network Interface Card
NLS	Native Language Support
NPTL	Native POSIX Threading Library
OF	Open Firmware
OSS	Open Sound System
PCH	Pre-Compiled Headers
PID	Process Identifier
PTY	pseudo terminal
QA	Quality Assurance
QOS	Quality Of Service
RAM	Random Access Memory
RPC	Remote Procedure Call
RTC	Real Time Clock
SCO	The Santa Cruz Operation
SATA	Serial ATA
SGR	Select Graphic Rendition
SHA1	Secure-Hash Algorithm 1
TLDP	The Linux Documentation Project
TFTP	Trivial File Transfer Protocol
TLS	Thread-Local Storage
UID	User Identifier
umask	user file-creation mask
USB	Universal Serial Bus
UTC	Coordinated Universal Time
UUID	Universally Unique Identifier
VC	Virtual Console

VGA Video Graphics Array

VT Virtual Terminal

Annexe B. Dépendances

Chaque paquet compilé dans CLFS dépend d'un ou plusieurs autres paquets afin de se compiler et de s'installer correctement. 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. A cause de ces dépendances, l'ordre dans lequel les paquets sont compilés dans CLFS est très important. Le but de cette page est de documenter les dépendances de chaque paquet compilé dans CLFS.

Pour chaque paquet qu'on compile, nous avons listé trois types de dépendances. Le premier liste les autres paquets qui doivent être disponibles afin de compiler et d'installer le paquet en question. Le second liste les paquets qui, en plus de ceux de la première liste, doivent être disponibles afin de lancer les suites de test. La dernière liste de dépendances contient les paquets qui exigent ce paquet pour être compilés et installés dans son 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/[binary] soient placés à l'intérieur de scripts installés dans le système final. Cela n'est évidemment pas souhaitable.

Autoconf

L'installation dépend de: Bash, Coreutils, Gawk, Grep, M4, Make, Perl, Sed et Texinfo

La suite de tests dépend de: Automake, Binutils, Diffutils, Findutils, GCC et Libtool

Doit être installé avant: Automake

Automake

L'installation dépend de: Autoconf, Bash, Binutils, Coreutils, Gawk, Grep, M4, Make, Perl, Sed et Texinfo

La suite de tests dépend de: Bison, Bzip2, DejaGNU, Diffutils, Expect, Findutils, Flex, GCC, Gettext, Gzip, Libtool, XZ-Utils et Tar. Peut aussi utiliser plusieurs autres paquets non installés dans CLFS.

Doit être installé avant: Aucun

Bash

L'installation dépend de: Bash, Bison, Coreutils, Diffutils, EGLIBC, Gawk, GCC, Grep, Make, Ncurses, Patch, Readline, Sed et Texinfo

La suite de tests dépend de: Aucun

Doit être installé avant: Aucun

Binutils

L'installation dépend de: Bash, Binutils, Coreutils, Diffutils, EGLIBC, File, Gawk, GCC, Grep, Make, Perl, Sed, Texinfo et Zlib

La suite de tests dépend de: DejaGNU et Expect

Doit être installé avant: Aucun

Bison

L'installation d'Ã©pend de:	Bash, Binutils, Coreutils, EGLIBC, Gawk, GCC, Grep, M4, Make et Sed
La suite de tests d'Ã©pend de:	Diffutils, Findutils et Gawk
Doit Ãªtre installÃ© avant:	Flex, Kbd et Tar

Bzip2

L'installation d'Ã©pend de:	Bash, Binutils, Coreutils, EGLIBC, GCC, Make
La suite de tests d'Ã©pend de:	Diffutils
Doit Ãªtre installÃ© avant:	Aucun

CLFS-Bootscripts

L'installation d'Ã©pend de:	Bash, Coreutils, Make et Sed
La suite de tests d'Ã©pend de:	Aucun
Doit Ãªtre installÃ© avant:	Aucun

CLooG-PPL

L'installation d'Ã©pend de:	Bash, Binutils, Coreutils, Diffutils, EGLIBC, Gawk, GCC, Grep, GMP, Make, MPC, MPFR, PPL, Sed et Texinfo
La suite de tests d'Ã©pend de:	Aucune
Doit Ãªtre installÃ© avant:	GCC

Coreutils

L'installation d'Ã©pend de:	Bash, Binutils, Coreutils, EGLIBC, Gawk, GCC, GMP, Grep, Make, Patch, Perl, Sed et Texinfo
La suite de tests d'Ã©pend de:	Diffutils, E2fsprogs, Findutils, Util-linux
Doit Ãªtre installÃ© avant:	Bash, Diffutils, Findutils, Man et Udev

DejaGNU

L'installation d'Ã©pend de:	Bash, Coreutils, Diffutils, GCC, Grep, Make et Sed
La suite de tests d'Ã©pend de:	Aucun
Doit Ãªtre installÃ© avant:	Aucun

DHCPD

L'installation d'Ã©pend de:	Bash, Coreutils, GCC, Make, Sed
La suite de tests d'Ã©pend de:	No testsuite available
Doit Ãªtre installÃ© avant:	Aucun

Diffutils

L'installation d'Ã©pend de:	Bash, Binutils, Coreutils, EGLIBC, GCC, Grep, Make, Patch, Sed et Texinfo
La suite de tests d'Ã©pend de:	No testsuite available
Doit Ãªtre installÃ© avant:	Aucun

EGLIBC

L'installation d'Ã©pend de:	Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Gettext, Grep, Gzip, Make, Perl, Sed et Texinfo
La suite de tests d'Ã©pend de:	Aucun
Doit Ãªtre installÃ© avant:	Aucun

Expect

L'installation d'Ã©pend de:	Bash, Binutils, Coreutils, Diffutils, EGLIBC, GCC, Grep, Make, Patch, Sed et Tcl
La suite de tests d'Ã©pend de:	Aucun
Doit Ãªtre installÃ© avant:	Aucun

E2fsprogs

L'installation d'Ã©pend de:	Bash, Binutils, Coreutils, EGLIBC, Gawk, GCC, Gettext, Grep, Gzip, Make, Pkg-config, Sed, Texinfo et Util-linux
La suite de tests d'Ã©pend de:	Bzip2 and Diffutils
Doit Ãªtre installÃ© avant:	Aucune

File

L'installation d'Ã©pend de:	Bash, Binutils, Coreutils, Diffutils, EGLIBC, Gawk, GCC, Grep, Make, Sed et Zlib
La suite de tests d'Ã©pend de:	No testsuite available
Doit Ãªtre installÃ© avant:	Aucun

Findutils

L'installation d'Ã©pend de:	Bash, Binutils, Coreutils, EGLIBC, GCC, Grep, Make, Sed et Texinfo
La suite de tests d'Ã©pend de:	DejaGNU, Diffutils et Expect
Doit Ãªtre installÃ© avant:	Aucun

Flex

L'installation d'Ã©pend de:	Bash, Binutils, Coreutils, EGLIBC, GCC, Grep, M4, Make, Sed et Texinfo
La suite de tests d'Ã©pend de:	Bison, Diffutils et Gawk
Doit Ãªtre installÃ© avant:	IPRoute2, Kbd et Man

Gawk

L'installation d'Ã©pend de:	Bash, Binutils, Coreutils, EGLIBC, GCC, Grep, Make, Sed et Texinfo
La suite de tests d'Ã©pend de:	Diffutils
Doit Ãªtre installÃ© avant:	Aucun

Gcc

L'installation d'Ã©pend de:	Bash, Binutils, Coreutils, Diffutils, EGLIBC, Findutils, Gawk, GCC, GMP, Grep, Make, MPFR, Patch, Perl, Sed, Tar et Texinfo
La suite de tests d'Ã©pend de:	DejaGNU et Expect
Doit Ãªtre installÃ© avant:	Aucun

Gettext

L'installation d'Ã©pend de:	Bash, Binutils, Coreutils, Diffutils, EGLIBC, Findutils, Gawk, GCC, Grep, Make, Sed et Texinfo
La suite de tests d'Ã©pend de:	Tar et Tcl
Doit Ãªtre installÃ© avant:	Automake

Glib

L'installation d'Ã©pend de:	bash, binutils, coreutils, gawk, gcc, gettext, make & M4.
La suite de tests d'Ã©pend de:	Unknown
Doit Ãªtre installÃ© avant:	Pkg-config

GMP

L'installation d'Ã©pend de:	Bash, Binutils, Coreutils, Diffutils, EGLIBC, Gawk, GCC, Grep, M4, Make, Sed et Texinfo
La suite de tests d'Ã©pend de:	Aucun
Doit Ãªtre installÃ© avant:	MPFR, GCC

Grep

L'installation d'Ã©pend de:	Bash, Binutils, Coreutils, EGLIBC, GCC, Grep, Make, Patch, Sed et Texinfo
La suite de tests d'Ã©pend de:	Diffutils et Gawk
Doit Ãªtre installÃ© avant:	Man

Groff

L'installation d'Ã©pend de:	Bash, Binutils, Coreutils, EGLIBC, Gawk, GCC, Grep, Make, Perl Sed et Texinfo
La suite de tests d'Ã©pend de:	No testsuite available
Doit Ãªtre installÃ© avant:	Man et Perl

Gzip

L'installation d'Ã©pend de:	Bash, Binutils, Coreutils, EGLIBC, GCC, Grep, Make, Sed et Texinfo
La suite de tests d'Ã©pend de:	Diffutils
Doit Ãªtre installÃ© avant:	Man

lana-Etc

L'installation d'Ã©pend de:	Coreutils, Gawk et Make
La suite de tests d'Ã©pend de:	No testsuite available
Doit Ãªtre installÃ© avant:	Perl

IProute2

L'installation d'Ã©pend de:	Bash, Binutils, Bison, Coreutils, EGLIBC, Findutils, Flex, GCC, Make, Linux-Headers et Sed
La suite de tests d'Ã©pend de:	No testsuite available
Doit Ãªtre installÃ© avant:	Aucun

IPutils

L'installation d'Ã©pend de:	Bash, Binutils, Coreutils, EGLIBC, GCC et Make
La suite de tests d'Ã©pend de:	No testsuite available
Doit Ãªtre installÃ© avant:	Aucun

Kbd

L'installation d'Ã©pend de:	Bash, Binutils, Coreutils, EGLIBC, Gawk, GCC, Gzip, Make et Sed
La suite de tests d'Ã©pend de:	No testsuite available
Doit Ãªtre installÃ© avant:	Aucun

Less

L'installation d'Ã©pend de:	Bash, Binutils, Coreutils, EGLIBC, GCC, Grep, Make, Ncurses et Sed
La suite de tests d'Ã©pend de:	No testsuite available
Doit Ãªtre installÃ© avant:	Aucun

Libee

L'installation d'Ã©pend de:	Bash, Binutils, Coreutils, Diffutils, EGLIBC, Findutils, Gawk, GCC, Grep, Libestr, Make, Pkg-config, Sed et Texinfo
La suite de tests d'Ã©pend de:	Aucune
Doit Ãªtre installÃ© avant:	Rsyslog

Libestr

L'installation d'Ã©pend de:	Bash, Binutils, Coreutils, Diffutils, EGLIBC, Findutils, Gawk, GCC, Grep, Make, Sed et Texinfo
La suite de tests d'Ã©pend de:	Aucune
Doit Ãªtre installÃ© avant:	Libee and Rsyslog

Libtool

L'installation d'Ã©pend de:	Bash, Binutils, Coreutils, Diffutils, EGLIBC, Findutils, Gawk, GCC, Grep, Make, Sed et Texinfo
La suite de tests d'Ã©pend de:	Autoconf
Doit Ãªtre installÃ© avant:	Aucun

Linux-Headers

L'installation d'Ã©pend de:	Binutils, Coreutils, Findutils, GCC, Grep, Make, Perl et Sed
La suite de tests d'Ã©pend de:	No testsuite available
Doit Ãªtre installÃ© avant:	Aucun

Linux Kernel

L'installation d'Ã©pend de:	Bash, Binutils, Coreutils, Diffutils, EGLIBC, Findutils, GCC, Grep, Gzip, Make, Module-Init-Tools, Ncurses, Perl et Sed
La suite de tests d'Ã©pend de:	No testsuite available
Doit Ãªtre installÃ© avant:	Aucun

M4

L'installation d'Ã©pend de:	Bash, Binutils, Coreutils, EGLIBC, Gawk, GCC, Grep, Make, Sed et Texinfo
La suite de tests d'Ã©pend de:	Diffutils
Doit Ãªtre installÃ© avant:	Autoconf et Bison

Make

L'installation d'Ã©pend de:	Bash, Binutils, Coreutils, EGLIBC, GCC, Grep, Make, Sed et Texinfo
La suite de tests d'Ã©pend de:	Perl et Procps
Doit Ãªtre installÃ© avant:	Aucun

Man

L'installation d'Ã©pend de:	Bash, Binutils, Bzip2, Coreutils, EGLIBC, Gawk, GCC, Grep, Groff, Gzip, Less, XZ-Utils, Make et Sed
La suite de tests d'Ã©pend de:	No testsuite available
Doit Ãªtre installÃ© avant:	Aucun

Man-Pages

L'installation d'Ã©pend de:	Bash, Coreutils et Make
La suite de tests d'Ã©pend de:	No testsuite available
Doit Ãªtre installÃ© avant:	Aucun

MPC

L'installation d'Ã©pend de:	Bash, Binutils, Coreutils, Diffutils, EGLIBC, Gawk, GCC, Grep, GMP, Make, MPFR, Sed et Texinfo
La suite de tests d'Ã©pend de:	Aucune
Doit Ãªtre installÃ© avant:	GCC

MPFR

L'installation d'Ã©pend de:	Bash, Binutils, Coreutils, Diffutils, EGLIBC, Gawk, GCC, Grep, GMP, Make, Sed et Texinfo
La suite de tests d'Ã©pend de:	Aucun
Doit Ãªtre installÃ© avant:	GCC

Module-Init-Tools

L'installation d'Ã©pend de:	Bash, Binutils, Coreutils, EGLIBC, Findutils, GCC, Grep, Make, Sed et Zlib
La suite de tests d'Ã©pend de:	Diffutils, File, Gawk et Gzip
Doit Ãªtre installÃ© avant:	Aucun

Ncurses

L'installation d'Ã©pend de:	Bash, Binutils, Coreutils, Diffutils, EGLIBC, Gawk, GCC, Grep, Make et Sed
La suite de tests d'Ã©pend de:	No testsuite available
Doit Ãªtre installÃ© avant:	Bash, GRUB, Inetutils, Less, Procps, Psmisc, Readline, Texinfo, Util-linux et Vim

Patch

L'installation d'Ã©pend de:	Bash, Binutils, Coreutils, EGLIBC, GCC, Grep, Make et Sed
La suite de tests d'Ã©pend de:	No testsuite available
Doit Ãªtre installÃ© avant:	Aucun

Perl

L'installation d'Ã©pend de:	Bash, Binutils, Coreutils, EGLIBC, Gawk, GCC, Grep, Make et Sed
La suite de tests d'Ã©pend de:	Gzip, Iana-Etc et Procps, Tar
Doit Ãªtre installÃ© avant:	Autoconf

Pkg-config

L'installation d'Ã©pend de:	Bash, Binutils, Coreutils, Diffutils, EGLIBC, Gawk, GCC, Grep, Make et Sed
La suite de tests d'Ã©pend de:	Aucune
Doit Ãªtre installÃ© avant:	Util-linux, E2fsprogs

PPL

L'installation d'Ã©pend de:	Bash, Binutils, Coreutils, Diffutils, EGLIBC, Gawk, GCC, Grep, GMP, Make, MPC, MPFR, Sed et Texinfo
La suite de tests d'Ã©pend de:	Aucune
Doit Ãªtre installÃ© avant:	GCC

Procps

L'installation d'Ã©pend de:	Bash, Binutils, Coreutils, EGLIBC, GCC, Make et Ncurses
La suite de tests d'Ã©pend de:	No testsuite available
Doit Ãªtre installÃ© avant:	Aucun

Psmisc

L'installation d'Ã©pend de:	Binutils, Coreutils, Diffutils, EGLIBC, Gawk, GCC, Grep, libee, Libestr, Make, Sed et Zlib
La suite de tests d'Ã©pend de:	No testsuite available
Doit Ãªtre installÃ© avant:	Aucun

Readline

L'installation d'Ã©pend de:	Bash, Binutils, Coreutils, EGLIBC, GCC, Grep, Make, Ncurses, Patch, Sed et Texinfo
La suite de tests d'Ã©pend de:	No testsuite available
Doit Ãªtre installÃ© avant:	Bash

Rsyslog

L'installation d'Ã©pend de:	Binutils, Coreutils, Diffutils, EGLIBC, Gawk, GCC, Grep, Make et Sed
La suite de tests d'Ã©pend de:	No testsuite available
Doit Ãªtre installÃ© avant:	Aucun

Sed

L'installation d'Ã©pend de:	Bash, Binutils, Coreutils, EGLIBC, GCC, Grep, Make, Sed et Texinfo
La suite de tests d'Ã©pend de:	Diffutils et Gawk
Doit Ãªtre installÃ© avant:	E2fsprogs, File, Libtool et Shadow

Shadow

L'installation d'Ã©pend de:	Bash, Binutils, Coreutils, Diffutils, EGLIBC, Findutils, Gawk, GCC, Gettext, Grep, Make et Sed
La suite de tests d'Ã©pend de:	No testsuite available
Doit Ãªtre installÃ© avant:	Aucun

Sysvinit

L'installation d'Ã©pend de:	Binutils, Coreutils, EGLIBC, GCC, Make et Sed
La suite de tests d'Ã©pend de:	No testsuite available
Doit Ãªtre installÃ© avant:	Aucun

Tar

L'installation d'Ã©pend de:	Bash, Binutils, Bison, Coreutils, EGLIBC, GCC, Grep, Make, Sed et Texinfo
La suite de tests d'Ã©pend de:	Diffutils, Findutils, Gawk et Gzip
Doit Ãªtre installÃ© avant:	Aucun

Tcl

L'installation d'Ã©pend de:	Bash, Binutils, Coreutils, Diffutils, EGLIBC, GCC, Grep, Make et Sed
La suite de tests d'Ã©pend de:	Aucun
Doit Ãªtre installÃ© avant:	Aucun

Texinfo

L'installation d'Ã©pend de:	Bash, Binutils, Coreutils, EGLIBC, Gawk, GCC, Grep, Make, Ncurses et Sed
La suite de tests d'Ã©pend de:	Diffutils et Gzip
Doit Ãªtre installÃ© avant:	Aucun

Udev

L'installation d'Ã©pend de:	Binutils, Coreutils, Diffutils, EGLIBC, Gawk, GCC, Grep, Make et Sed
La suite de tests d'Ã©pend de:	No testsuite available
Doit Ãªtre installÃ© avant:	Aucun

Util-linux

L'installation d'Ã©pend de:	Bash, Binutils, Coreutils, EGLIBC, GCC, Grep, Make, Ncurses, Pkg-config, Sed, Texinfo et Zlib
La suite de tests d'Ã©pend de:	No testsuite available
Doit Ãªtre installÃ© avant:	E2fsprogs

Vim

L'installation d'Ã©pend de:	Bash, Binutils, Coreutils, Diffutils, EGLIBC, Findutils, Gawk, GCC, Gettext, Grep, Make, Ncurses, Perl et Sed
La suite de tests d'Ã©pend de:	Gzip
Doit Ãªtre installÃ© avant:	Aucun

XZ-Utils

L'installation d'Ã©pend de:	Bash, Binutils, Coreutils, Diffutils, EGLIBC, Findutils, Gawk, GCC, Grep, Make et Sed
La suite de tests d'Ã©pend de:	Aucun
Doit Ãªtre installÃ© avant:	Aucun

Zlib

L'installation d'Ã©pend de:	Bash, Binutils, Coreutils, EGLIBC, GCC, Make et Sed
La suite de tests d'Ã©pend de:	Aucun
Doit Ãªtre installÃ© avant:	File, Module-Init-Tools et Util-linux

Annexe C. x86 Dependencies

This page contains dependency information for packages specific to x86.

GRUB2

L'installation d'Ã©pend de:	Bash, Binutils, Bison Coreutils, Diffutils, EGLIBC, Gawk, GCC, Gettext, Grep, Make, Ncurses, Sed et Texinfo
La suite de tests d'Ã©pend de:	None
Doit Ãªtre installÃ© avant:	None

Annexe D. Raison de la présence des paquets

CLFS comprend de nombreux paquets, parmi lesquels certains pourraient ne pas être obligatoires pour un système "minimal", mais ils n'en demeurent pas moins très utiles. L'objectif de cette page est de lister les raisons de la présence de chaque paquet dans le livre.

- Autoconf

Le paquet Autoconf contient des programmes produisant des scripts shell qui configurent automatiquement le code source. Ceci sert aux développeurs de logiciels et à tous ceux qui veulent installer des paquets non fournis avec des scripts configure, tels que certains paquets de CBLFS.

- Automake

Le paquet Automake contient des programmes pour générer des Makefiles utilisables avec Autoconf. Cela peut servir aux développeurs logiciels.

- Bash

Ce paquet contient le shell Bourne-Again SHell. Le shell est un composant important du système Linux, car il doit exister un moyen de permettre aux utilisateurs d'entrer des commandes.

- Binutils

Ce paquet contient des programmes pour gérer des fichiers objets. Les programmes de ce paquet sont nécessaires pour compiler la plupart des paquets de CLFS.

- Bison

Ce paquet contient des programmes requis par plusieurs paquets de CLFS

- Bzip2

Les programmes de ce paquet servent à compresser des fichiers pour diminuer leur taille. Ils servent aussi à décompresser les archives tar de nombreux paquets CLFS.

- CLFS-Bootscripts

Ce paquet contient un certain nombre de scripts qui démarrent au moment de l'amorçage et qui effectuent des tâches essentielles comme le montage/vérification des systèmes de fichiers et le lancement de l'interface réseau.

- CLooG-PPL

Ce paquet est utilisé par GCC

- Coreutils

Ce paquet contient de nombreuses outils de base en ligne de commande pour gérer des fichiers, nécessaire pour l'installation de chaque paquet de CLFS.

- DejaGNU

Ce paquet est nécessaire pour la suite de tests de plusieurs paquets, surtout GCC et Binutils.

- DHCPD

Ce paquet permet une configuration automatique des interfaces réseaux à partir d'un serveur DHCP. Lui ou certains paquets offrant un client DHCP sont requis pour se connecter à un serveur DHCP.

- Diffutils

Ce paquet contient des programmes pour comparer des fichiers et il peut être aussi utilisé pour créer des correctifs. Il est requis par les procédures d'installation de nombreux paquets CLFS.

- EGLIBC

Tout programme C lié de façon dynamique (ce qui est le cas de presque tout dans CLFS) a besoin d'une bibliothèque C pour se compiler et se lancer.

- Expect

Ce paquet est nécessaire aux suites de tests de nombreux paquets.

- E2fsprogs

Les programmes de ce paquet sont utilisés pour créer et maintenir des systèmes de fichiers ext2/3/4.

- File

Ce paquet contient un programme qui détermine le type d'un fichier donné. Il est nécessaire pour certains paquets CLFS.

- Findutils

Ce paquet contient des programmes pour chercher des fichiers à partir de certains critères et, éventuellement, y appliquer des commandes. Il est utilisé par les procédures d'installation de nombreux paquets CLFS.

- Flex

Ce paquet contient un outil pour générer des analyseurs de texte. Il est utilisé par plusieurs paquets de CLFS

- Gawk

Ce paquet contient des programmes pour manipuler des fichiers texte en utilisant le langage AWK. Il est utilisé par les procédures d'installation de nombreux paquets dans CLFS.

- Gcc

Ce paquet contient un compilateur C nécessaire pour compiler la plupart des paquets de CLFS.

- Gettext

Outil permettant aux programmeurs d'implémenter facilement l'internationalisation dans leurs programmes. C'est une dépendance requise pour un certain nombre de paquets

- Glib

Requis pour pkg-config

- GMP

Ce paquet est requis par GCC

- Grep

Ce paquet contient des programmes pour chercher du texte dans des fichiers. Exigé par de nombreux paquets dans CLFS.

- Groff

Ce paquet est requis par Man

- Gzip

Sert à compresser des fichiers pour économiser de la place. Il sert aussi à décompresser les archives tar de nombreux paquets CLFS

- Iana-Etc

Ce paquet fournit les fichiers `/etc/services` et `/etc/protocols`. Ces fichiers relient des noms de services à des numéros de ports ainsi que des noms de protocoles à leur numéro de ports correspondants. Ces fichiers sont essentiels pour que de nombreux programmes basés sur le réseau fonctionnent correctement.

- IProute2

Ce paquet contient des programmes d'administration des interfaces réseaux.

- IPutils

Ce paquet contient plusieurs outils de gestion de base du réseau.

- Kbd

Contient les fichiers de tables de touches et des outils claviers compatibles avec le noyau Linux.

- Kmod

Ce paquet contient des programmes aidant à charger et décharger des modules du noyau.

- Less

Un programme vous permettant de visualiser des fichiers textes page par page. Utilisé par Man pour afficher des pages de man.

- Libee

Ce paquet contient une bibliothèque d'expression d'événements. Il est nécessaire pour Rsyslog.

- Libestr

Ce paquet contient une bibliothèque de chaînes essentielles. Il est nécessaire pour Rsyslog.

- Libtool

Le paquet Libtool contient le script de support de la bibliothèque générique GNU. Il est utilisé par certains paquets CLFS.

- Linux-Headers

Ce paquet contient des en-têtes récupérées du noyau Linux..Ces en-têtes sont exigées pour que Glibc compile.

- Noyau Linux

Le système d'exploitation Linux.

- M4

Ce paquet contient un processeur de macros. Il est exigé par plusieurs paquets de CLFS, notamment Bison.

- Make

Nécessaire pour l'installation de la plupart des paquets de CLFS

- Man

Utilisé pour visualiser des pages de man

- Man-Pages

Un certain nombre de pages de man utiles et non fournies par d'autres paquets

- MPC

Ce paquet est requis par GCC

- MPFR

Ce paquet est requis par GCC

- Ncurses

Nécessaire pour plusieurs paquets de CLFS tels que Vim, Bash, e Less

- Patch

Utilisé pour appliquer des correctifs dans plusieurs paquets CLFS

- Perl

Le paquet Perl contient le *Practical Extraction and Report Language* (langage pratique de rapport et d'extraction). Il est exigé par plusieurs paquets CLFS.

- Pkg-config

Exigé par E2fsprogs

- PPL

Ce paquet est utilisé par GCC

- Procps

Fournit un certain nombre de petits outils simples qui donnent des informations sur le système de fichiers / `proc`.

- Psmisc

Fournit encore plus d'outils donnant des informations sur le système de fichiers / `proc`.

- Readline

La bibliothèque Readline fournit un ensemble de fonctions qu'utilisent les applications permettant aux utilisateurs d'éditer des lignes de commande au moment où ils les écrivent. Il est essentiel pour que des programmes d'entrée comme **bash** fonctionnent correctement.

- Rsyslog

Rsyslog est un syslogd multi-threadé amélioré qui supporte plusieurs fondations avec très peu de dépendances. Il fournit un programme qui enregistre divers événements systèmes dans les fichiers de `/var/log`.

- Sed

Ce paquet contient un éditeur de flux. Il est utilisé dans les procédures d'installation de la plupart des paquets CLFS.

- Shadow

Ce paquet contient des programmes aidant à administrer des utilisateurs, des groupes et des mots de passe.

- Sysvinit

Sysvinit est le démon d'initialisation avec lequel fonctionnent les scripts de démarrage écrits pour clfs.

- Tar

Exigé pour déballer les archives tar, là où toutes les archives CLFS sont distribuées

- Tcl

Requis pour les suites de tests de plusieurs paquets

- Texinfo

Ce paquet contient des programmes pour visualiser, installer convertir des pages info. Il est utilisé dans les procédures d'installation de nombreux paquets CLFS.

- Udev

Le paquet Udev contient des programmes de création dynamiques de nœuds de périphériques.

- Util-linux

Le paquet Util-linux contient des programmes généralistes. Figurent parmi eux des outils de gestion des systèmes de fichiers, des consoles, des partitions et des messages. Il comprend aussi des bibliothèques exigées par E2fsprogs.

- Vim

Le paquet Vim contient un éditeur de texte. Les utilisateurs peuvent le remplacer par Nano, Joe, Emacs, ou autre éditeur préféré.

- XZ-Utills

Sert à compresser des fichiers pour diminuer leur taille. Nécessaire aussi pour décompresser des archives tar de nombreux paquets CLFS

- Zlib

Le paquet Zlib contient des routines de compression et de décompression utilisés par certains programmes.

Annexe E. Open Publication License

v1.0, 8 June 1999

I. REQUIREMENTS ON BOTH UNMODIFIED AND MODIFIED VERSIONS

The Open Publication works may be reproduced and distributed in whole or in part, in any medium physical or electronic, provided that the terms of this license are adhered to, and that this license or an incorporation of it by reference (with any options elected by the author(s) and/or publisher) is displayed in the reproduction.

Proper form for an incorporation by reference is as follows:

Copyright © <year> by <author's name or designee>. This material may be distributed only subject to the terms and conditions set forth in the Open Publication License, vX.Y or later (the latest version is presently available at <http://www.opencontent.org/openpub/>).

The reference must be immediately followed with any options elected by the author(s) and/or publisher of the document (see section VI).

Commercial redistribution of Open Publication-licensed material is permitted.

Any publication in standard (paper) book form shall require the citation of the original publisher and author. The publisher and author's names shall appear on all outer surfaces of the book. On all outer surfaces of the book the original publisher's name shall be as large as the bridgehead of the work and cited as possessive with respect to the bridgehead.

II. COPYRIGHT

The copyright to each Open Publication is owned by its author(s) or designee.

III. SCOPE OF LICENSE

The following license terms apply to all Open Publication works, unless otherwise explicitly stated in the document.

Mere aggregation of Open Publication works or a portion of an Open Publication work with other works or programs on the same media shall not cause this license to apply to those other works. The aggregate work shall contain a notice specifying the inclusion of the Open Publication material and appropriate copyright notice.

SEVERABILITY. If any part of this license is found to be unenforceable in any jurisdiction, the remaining portions of the license remain in force.

NO WARRANTY. Open Publication works are licensed and provided "as is" without warranty of any kind, express or implied, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose or a warranty of non-infringement.

IV. REQUIREMENTS ON MODIFIED WORKS

All modified versions of documents covered by this license, including translations, anthologies, compilations and partial documents, must meet the following requirements:

1. The modified version must be labeled as such.
2. The person making the modifications must be identified and the modifications dated.
3. Acknowledgement of the original author and publisher if applicable must be retained according to normal academic citation practices.

4. The location of the original unmodified document must be identified.
5. The original author's (or authors') name(s) may not be used to assert or imply endorsement of the resulting document without the original author's (or authors') permission.

V. GOOD-PRACTICE RECOMMENDATIONS

In addition to the requirements of this license, it is requested from and strongly recommended of redistributors that:

1. If you are distributing Open Publication works on hardcopy or CD-ROM, you provide email notification to the authors of your intent to redistribute at least thirty days before your manuscript or media freeze, to give the authors time to provide updated documents. This notification should describe modifications, if any, made to the document.
2. All substantive modifications (including deletions) be either clearly marked up in the document or else described in an attachment to the document.
3. Finally, while it is not mandatory under this license, it is considered good form to offer a free copy of any hardcopy and CD-ROM expression of an Open Publication-licensed work to its author(s).

VI. LICENSE OPTIONS

The author(s) and/or publisher of an Open Publication-licensed document may elect certain options by appending language to the reference to or copy of the license. These options are considered part of the license instance and must be included with the license (or its incorporation by reference) in derived works.

A. To prohibit distribution of substantively modified versions without the explicit permission of the author(s). "Substantive modification" is defined as a change to the semantic content of the document, and excludes mere changes in format or typographical corrections.

To accomplish this, add the phrase 'Distribution of substantively modified versions of this document is prohibited without the explicit permission of the copyright holder.' to the license reference or copy.

B. To prohibit any publication of this work or derivative works in whole or in part in standard (paper) book form for commercial purposes is prohibited unless prior permission is obtained from the copyright holder.

To accomplish this, add the phrase 'Distribution of the work or derivative of the work in any standard (paper) book form is prohibited unless prior permission is obtained from the copyright holder.' to the license reference or copy.

OPEN PUBLICATION POLICY APPENDIX

(This is not considered part of the license.)

Open Publication works are available in source format via the Open Publication home page at <http://works.opencontent.org/>.

Open Publication authors who want to include their own license on Open Publication works may do so, as long as their terms are not more restrictive than the Open Publication license.

If you have questions about the Open Publication License, please contact David Wiley at dw@opencontent.org, and/or the Open Publication Authors' List at opal@opencontent.org, via email.

To **subscribe** to the Open Publication Authors' List: Send E-mail to opal-request@opencontent.org with the word "subscribe" in the body.

To **post** to the Open Publication Authors' List: Send E-mail to opal@opencontent.org or simply reply to a previous post.

To **unsubscribe** from the Open Publication Authors' List: Send E-mail to opal-request@opencontent.org with the word "unsubscribe" in the body.

Index

Paquets

Autoconf: 205
 Automake: 206
 Bash: 208
 système temporaire: 66
 Binutils: 153
 outils croisés: 44
 système temporaire: 62
 Bison: 191
 32 Bit: 190
 système temporaire: 68
 Bootscripts: 265
 démarrage: 107
 utilisation: 267
 Bzip2: 211
 32 bits: 210
 système temporaire: 69
 CLooG: 150
 32 Bit: 149
 cross-tools: 43
 temporary system: 60
 Coreutils: 183
 système temporaire: 70
 DejaGNU: 121
 DHCPD: 281
 Diffutils: 213
 système temporaire: 72
 E2fsprogs: 177
 32 bit: 176
 démarrage: 94
 EGLIBC: 132
 32 bit: 129
 outils croisés, 32 bit: 48
 outils croisés, 64 bits: 50
 Expect: 120
 File: 215
 32 bits: 214
 cross-tools: 36
 système temporaire: 74
 Findutils: 217
 système temporaire: 73
 Flex: 195
 32 bits: 194
 système temporaire: 75
 Gawk: 216
 système temporaire: 76
 GCC: 156
 outils croisés, final: 52
 outils croisés, statiques: 46
 système temporaire: 63
 Gettext: 219
 32 bit: 218
 système temporaire: 77
 GMP: 140
 32 Bit: 139
 cross-tools: 39
 système temporaire: 56
 Grep: 221
 système temporaire: 78
 Groff: 222
 GRUB: 260
 boot: 104
 bootable: 286
 configuring: 261
 Gzip: 226
 système temporaire: 79
 Iana-Etc: 188
 IPRoute2: 196
 IPutils: 227
 Kbd: 228
 Kmod: 237
 32 Bit: 236
 boot: 97
 Less: 225
 Libee: 244
 32 Bit: 243
 Libestr: 242
 32 Bit: 241
 Libtool: 193
 32 bit: 192
 Linux: 284
 démarrage: 102
 Linux-Headers: 127
 outils croisés: 35
 M4: 189
 système temporaire: 80
 temporary system: 37
 Make: 230
 système temporaire: 81
 Man: 234
 Man-pages: 128
 MPC: 145
 32 Bit: 144
 cross-tools: 41
 temporary system: 58
 MPFR: 143
 32 Bit: 142

- cross-tools: 40
- système temporaire: 57
- Multiarch Wrapper: 158
- Ncurses: 164
 - 32 bits: 162
- cross-tools: 38
- système temporaire: 65
- Patch: 239
 - système temporaire: 82
- Perl: 200
 - 32 bits: 198
- outils temporaires: 126
- Pkg-config: 167
- PPL: 147
 - 32 Bit: 146
- cross-tools: 42
- système temporaire: 59
- Procps: 174
 - 32 bits: 173
- Psmisc: 240
- Readline: 204
 - 32 bits: 203
- rsyslog: 245
 - configuration: 246
- Sed: 161
 - système temporaire: 83
- Shadow: 180
 - boot: 93
 - configuration: 181
- Sysvinit: 248
 - configuration: 248
 - démarrage: 95
 - démarrage, configuration: 95
- Tar: 251
 - système temporaire: 84
- Tcl: 119
- Texinfo: 252
 - système temporaire: 85
- Udev: 255
 - 32 Bits: 254
 - démarrage: 98
 - utilisation: 269
- Util-linux: 169
 - 32 bits: 168
 - chroot: 110
 - démarrage: 92
- Vim: 257
 - système temporaire: 86
- XZ-Uutils: 232
 - 32 bits: 231

- système temporaire: 88
- Zlib: 152
 - 32 Bit: 151
- démarrage: 61

Programmes

- a2p: 200, 201
- acinstall: 206, 206
- aclocal: 206, 206
- aclocal-1.12: 206, 206
- addftinfo: 222, 222
- addpart: 169, 170
- addr2line: 153, 154
- afmtodit: 222, 222
- agetty: 169, 170
- apropos: 234, 235
- ar: 153, 154
- arch: 169, 170
- as: 153, 154
- ata_id: 255, 256
- autoconf: 205, 205
- autoheader: 205, 205
- autom4te: 205, 205
- automake: 206, 206
- automake-1.12: 206, 206
- autopoint: 219, 219
- autoreconf: 205, 205
- autoscan: 205, 205
- autoupdate: 205, 205
- awk: 216, 216
- badblocks: 177, 178
- base64: 183, 184
- basename: 183, 184
- bash: 208, 208
- bashbug: 208, 208
- bigram: 217, 217
- bison: 191, 191
- blkid: 169, 170
- blockdev: 169, 170
- bootlogd: 248, 249
- bunzip2: 211, 211
- bzcat: 211, 211
- bzcmp: 211, 211
- bzdiff: 211, 212
- bzegrep: 211, 212
- bzfgrep: 211, 212
- bzgrep: 211, 212
- bzip2: 211, 212
- bzip2recover: 211, 212
- bzless: 211, 212

bzmore: 211, 212	cpanp-run-perl: 200, 201
c++: 156, 157	cpp: 156, 157
c++filt: 153, 154	create_floppy_devices: 255, 256
c2ph: 200, 201	csplit: 183, 184
cal: 169, 170	ctrlaltdel: 169, 170
captoinfo: 164, 165	ctstat: 196, 196
cat: 183, 184	cut: 183, 184
catchsegv: 132, 136	cytune: 169, 170
cc: 156, 157	dasd_id: 255, 256
cdrom_id: 255, 256	date: 183, 184
cfdisk: 169, 170	dd: 183, 185
chage: 180, 181	ddate: 169, 170
chattr: 177, 178	deallocvt: 228, 228
chcon: 183, 184	debugfs: 177, 178
chcpu: 169, 170	delpart: 169, 170
chem: 222, 222	depcomp: 206, 206
chfn: 180, 181	depmod: 237, 237
chgpasswd: 180, 181	df: 183, 185
chgrp: 183, 184	diff: 213, 213
chmod: 183, 184	diff3: 213, 213
chown: 183, 184	dir: 183, 185
chpasswd: 180, 181	dircolors: 183, 185
chroot: 183, 184	dirname: 183, 185
chrt: 169, 170	dmesg: 169, 170, 169, 170
chsh: 180, 181	du: 183, 185
chvt: 228, 228	dumpe2fs: 177, 178
cksum: 183, 184	dumpkeys: 228, 228
clear: 164, 165	e2freefrag: 177, 178
clfskernel-[linux-version]: 284, 285	e2fsck: 177, 178
clockdiff: 227, 227	e2image: 177, 178
cloog: 150, 150	e2initrd_helper: 177, 178
cmp: 213, 213	e2label: 177, 178
code: 217, 217	e2undo: 177, 178
col: 169, 170	e4defrag: 177, 178
colcrt: 169, 170	echo: 183, 185
colrm: 169, 170	edd_id: 255, 256
column: 169, 170	efm_filter.pl: 257, 258
comm: 183, 184	efm_perl.pl: 257, 258
compile: 206, 206	egrep: 221, 221
compile_et: 177, 178	elfedit: 153, 154
config.charset: 219, 219	elisp-comp: 206, 206
config.guess: 206, 206	enc2xs: 200, 201
config.rpath: 219, 219	env: 183, 185
config.sub: 206, 206	envsubst: 219, 219
config_data: 200, 201	eqn: 222, 222
corelist: 200, 201	eqn2graph: 222, 222
cp: 183, 184	ex: 257, 259
cpan: 200, 201	expand: 183, 185
cpan2dist: 200, 201	expect: 120, 120
cpanp: 200, 201	expiry: 180, 181

expr:	183, 185	gpasswd:	180, 181
factor:	183, 185	gprof:	153, 154
faillog:	180, 181	grap2graph:	222, 223
fallocate:	169, 170	grcat:	216, 216
false:	183, 185	grep:	221, 221
fdformat:	169, 170	grn:	222, 223
fdisk:	169, 170	grodvi:	222, 223
fgconsole:	228, 228	groff:	222, 223
fgrep:	221, 221	groffer:	222, 223
file:	215, 215	grog:	222, 223
filefrag:	177, 178	grolbp:	222, 223
find:	217, 217	grolj4:	222, 223
find2perl:	200, 201	grops:	222, 223
findfs:	169, 170	grotty:	222, 223
findmnt:	169, 170	groupadd:	180, 181
firmware.sh:	255, 256	groupdel:	180, 181
flex:	195, 195	groupmems:	180, 181
flex++:	195, 195	groupmod:	180, 182
flock:	169, 170	groups:	183, 185
fmt:	183, 185	grpck:	180, 182
fold:	183, 185	grpconv:	180, 182
frcode:	217, 217	grpunconv:	180, 182
free:	174, 174	grub:	260, 262
fsck:	169, 170	grub-install:	260, 262
fsck.cramfs:	169, 170	grub-md5-crypt:	260, 262
fsck.ext2:	177, 178	grub-set-default:	260, 262
fsck.ext3:	177, 178	grub-terminfo:	260, 262
fsck.ext4:	177, 178	gtbl:	222, 223
fsck.ext4dev:	177, 178	gunzip:	226, 226
fsck.minix:	169, 170	gzexe:	226, 226
fsfreeze:	169, 170	gzip:	226, 226
fstab-decode:	248, 249	h2ph:	200, 202
fstrim:	169, 170	h2xs:	200, 202
fuser:	240, 240	halt:	248, 249
g++:	156, 157	head:	183, 185
gawk:	216, 216	hexdump:	169, 170
gawk-4.0.1:	216, 216	hostid:	183, 185
gcc:	156, 157	hostname:	183, 185
gcov:	156, 157	hostname:	219, 219
gdifflmk:	222, 222	hpftodit:	222, 223
gencat:	132, 136	hwclock:	169, 170
genl:	196, 196	iconv:	132, 136
geqn:	222, 222	iconvconfig:	132, 136
getconf:	132, 136	id:	183, 185
getent:	132, 136	ifcfg:	196, 197
getkeycodes:	228, 228	ifnames:	205, 205
getopt:	169, 170	ifstat:	196, 197
gettext:	219, 219	igawk:	216, 216
gettext.sh:	219, 219	indxbib:	222, 223
gettextize:	219, 219	info:	252, 252

infocmp: 164, 165	locate: 217, 217
infokey: 252, 252	logger: 169, 171
infotocap: 164, 165	login: 180, 182
init: 248, 249	logname: 183, 185
insmod: 237, 237	logoutd: 180, 182
install: 183, 185	logsave: 177, 178
install-info: 252, 252	look: 169, 171
install-sh: 206, 206	lookbib: 222, 223
instmodsh: 200, 202	losetup: 169, 171
ionice: 169, 171	ls: 183, 185
ip: 196, 197	lsattr: 177, 178
ipcmk: 169, 171	lsblk: 169, 171
ipcrm: 169, 171	lscpu: 169, 171
ipcs: 169, 171	lslocks: 169, 171
isosize: 169, 171	lsmod: 237, 237
join: 183, 185	lzcat: 232, 232
json_pp: 200, 202	lzcmp: 232, 232
kbdinfo: 228, 228	lzdiff: 232, 232
kbdrate: 228, 228	lzegrep: 232, 232
kbd_mode: 228, 228	lzfgrep: 232, 232
kill: 169, 171	lzgrep: 232, 232
killall: 240, 240	lzless: 232, 232
killall5: 248, 249	lzma: 232, 232
kmod: 237, 237	lzmadec: 232, 232
last: 248, 249	lzmore: 232, 232
lastb: 248, 250	m4: 189, 189
lastlog: 180, 182	make: 230, 230
ld: 153, 154	makedb: 132, 136
ld.bfd: 153, 154	makeinfo: 252, 252
ldattach: 169, 171	makewhatis: 234, 235
ldconfig: 132, 136	man: 234, 235
ldd: 132, 136	man2dvi: 234, 235
lddlibc4: 132, 136	man2html: 234, 235
less: 225, 225	mapscrn: 228, 228
less.sh: 257, 259	mbchk: 260, 262
lessecho: 225, 225	mcookie: 169, 171
lesskey: 225, 225	md5sum: 183, 185
lex: 195, 195	mdate-sh: 206, 206
libee-convert: 244, 244	mesg: 248, 250
libnetcfg: 200, 202	missing: 206, 206
libtool: 193, 193	mkdir: 183, 185
libtoolize: 193, 193	mke2fs: 177, 178
link: 183, 185	mkfifo: 183, 185
lkbib: 222, 223	mkfs: 169, 171
ln: 183, 185	mkfs.bfs: 169, 171
lnstat: 196, 197	mkfs.cramfs: 169, 171
loadkeys: 228, 228	mkfs.ext2: 177, 178
loadunimap: 228, 228	mkfs.ext3: 177, 178
locale: 132, 136	mkfs.ext4: 177, 178
localedef: 132, 136	mkfs.ext4dev: 177, 178

mkfs.minix: 169, 171	passwd: 180, 182
mkinstalldirs: 206, 206	paste: 183, 186
mklost+found: 177, 178	patch: 239, 239
mknod: 183, 185	pathchk: 183, 186
mkswap: 169, 171	path_id: 255, 256
mk_cmds: 177, 178	pcprofiledump: 132, 136
mmroff: 222, 223	pdfroff: 222, 223
modinfo: 237, 237	pdftexi2dvi: 252, 252
modprobe: 237, 237	peekfd: 240, 240
more: 169, 171	perl: 200, 202
mount: 169, 171	perl5.16.2: 200, 202
mountpoint: 169, 171	perlbug: 200, 202
msgattrib: 219, 219	perldoc: 200, 202
msgcat: 219, 219	perlvp: 200, 202
msgcmp: 219, 219	perlthanks: 200, 202
msgcomm: 219, 220	pfbtops: 222, 223
msgconv: 219, 220	pg: 169, 171
msgen: 219, 220	pgawk: 216, 216
msgexec: 219, 220	pgawk-4.0.1: 216, 216
msgfilter: 219, 220	pgrep: 174, 174
msgfmt: 219, 220	pic: 222, 223
msggrep: 219, 220	pic2graph: 222, 223
msginit: 219, 220	piconv: 200, 202
msgmerge: 219, 220	pidof: 248, 250
msgunfmt: 219, 220	ping: 227, 227
msguniq: 219, 220	pinky: 183, 186
mtrace: 132, 136	pivot_root: 169, 171
multiarch_wrapper: 158, 160	pkg-config: 167, 167
mv: 183, 185	pkill: 174, 174
mve.awk: 257, 259	pl2pm: 200, 202
namei: 169, 171	pldd: 132, 136
ncursesw5-config: 164, 165	pltags.pl: 257, 259
neqn: 222, 223	pmap: 174, 174
newgrp: 180, 182	pod2html: 200, 202
newusers: 180, 182	pod2latex: 200, 202
ngettext: 219, 220	pod2man: 200, 202
nice: 183, 185	pod2text: 200, 202
nl: 183, 185	pod2usage: 200, 202
nm: 153, 154	podchecker: 200, 202
nohup: 183, 185	podselect: 200, 202
nologin: 180, 182	post-grohtml: 222, 223
nproc: 183, 186	poweroff: 248, 250
nroff: 222, 223	ppl-config: 147, 148
nscd: 132, 136	ppl_lcdd: 147, 148
nstat: 196, 197	ppl_pips: 147, 148
objcopy: 153, 154	pr: 183, 186
objdump: 153, 154	pre-grohtml: 222, 223
od: 183, 186	preconv: 222, 223
openvt: 228, 229	printenv: 183, 186
partx: 169, 171	printf: 183, 186

prlimit: 169, 171	roff2text: 222, 224
prove: 200, 202	roff2x: 222, 224
prtstat: 240, 240	route: 196, 197
ps: 174, 174	routel: 196, 197
psed: 200, 202	rpcgen: 132, 136
psfaddtable: 228, 229	rsyslogd: 245, 247
psfgettable: 228, 229	rtacct: 196, 197
psfstripletable: 228, 229	rtcwake: 169, 171
psfxtable: 228, 229	rtmon: 196, 197
pstree: 240, 240	rtpr: 196, 197
pstree.x11: 240, 240	rtstat: 196, 197
pstruct: 200, 202	runcon: 183, 186
ptar: 200, 202	runlevel: 248, 250
ptardiff: 200, 202	runtest: 121, 121
ptargrep: 200, 202	rview: 257, 259
ptx: 183, 186	rvim: 257, 259
pt_chown: 132, 136	s2p: 200, 202
pwcat: 216, 216	script: 169, 171
pwck: 180, 182	scriptreplay: 169, 171
pwconv: 180, 182	scsi_id: 255, 256
pwd: 183, 186	sdiff: 213, 213
pwdx: 174, 175	sed: 161, 161
pwunconv: 180, 182	seq: 183, 186
py-compile: 206, 207	setarch: 169, 172
ranlib: 153, 154	setfont: 228, 229
raw: 169, 171	setkeycodes: 228, 229
rdisc: 227, 227	setleds: 228, 229
readelf: 153, 154	setmetamode: 228, 229
readlink: 183, 186	setsid: 169, 172
readprofile: 169, 171	setterm: 169, 172
realpath: 183, 186	setvtrgb: 228, 229
reboot: 248, 250	sfdisk: 169, 172
recode-sr-latin: 219, 220	sg: 180, 182
ref: 257, 259	sh: 208, 209
refer: 222, 223	sha1sum: 183, 186
rename: 169, 171	sha224sum: 183, 186
renice: 169, 171	sha256sum: 183, 186
reset: 164, 165	sha384sum: 183, 186
resize2fs: 177, 178	sha512sum: 183, 186
resizecons: 228, 229	shasum: 200, 202
resizepart: 169, 171	showconsolefont: 228, 229
rev: 169, 171	showkey: 228, 229
rm: 183, 186	shred: 183, 186
rmdir: 183, 186	shtags.pl: 257, 259
rmmod: 237, 238	shuf: 183, 186
rmt: 251, 251	shutdown: 248, 250
roff2dvi: 222, 223	size: 153, 154
roff2html: 222, 224	skill: 174, 175
roff2pdf: 222, 224	slabtop: 174, 175
roff2ps: 222, 224	sleep: 183, 186

sln: 132, 136	tracepath: 227, 227
snice: 174, 175	tracepath6: 227, 227
soelim: 222, 224	traceroute6: 227, 227
sort: 183, 186	troff: 222, 224
sotrust: 132, 137	true: 183, 187
splain: 200, 202	truncate: 183, 187
split: 183, 186	tset: 164, 165
sprof: 132, 137	tsort: 183, 187
ss: 196, 197	tty: 183, 187
stat: 183, 186	tune2fs: 177, 178
stdbuf: 183, 186	tunelp: 169, 172
strings: 153, 154	tzselect: 132, 137
strip: 153, 155	udevadm trigger: 255, 256
stty: 183, 186	udevadm: 255, 255
su: 180, 182	udevadm control: 255, 255
sulogin: 169, 172	udevadm monitor: 255, 255
sum: 183, 186	udevadm test: 255, 256
swapon: 169, 172	udevmd: 255, 255
swaplabel: 169, 172	udevinfo: 255, 255
swapoff: 169, 172	udevsettle: 255, 255
swapon: 169, 172	ul: 169, 172
switch_root: 169, 172	umount: 169, 172
symlink-tree: 206, 207	uname: 183, 187
sync: 183, 186	uncompress: 226, 226
sysctl: 174, 175	unexpand: 183, 187
tabs: 164, 165	unicode_start: 228, 229
tac: 183, 186	unicode_stop: 228, 229
tail: 183, 186	uniq: 183, 187
tailf: 169, 172	unlink: 183, 187
tar: 251, 251	unlzma: 232, 232
taskset: 169, 172	unshare: 169, 172
tbl: 222, 224	unxz: 232, 232
tc: 196, 197	updatedb: 217, 217
tcsh: 119, 119	uptime: 174, 175
tcsh-version: 119, 119	usb_id: 255, 256
tcsh: 119, 119	useradd: 180, 182
tcsh: 119, 119	userdel: 180, 182
tee: 183, 186	usermod: 180, 182
telinit: 248, 250	users: 183, 187
test: 183, 186	utmpdump: 169, 172
texi2dvi: 252, 252	uudd: 169, 172
texi2pdf: 252, 253	uuidgen: 169, 172, 169, 172
texindex: 252, 253	v4l_id: 255, 256
tfmtodit: 222, 224	vdir: 183, 187
tic: 164, 165	vi: 257, 259
timeout: 183, 187	view: 257, 259
tload: 174, 175	vigr: 180, 182
toe: 164, 165	vim: 257, 259
top: 174, 175	vim132: 257, 259
touch: 183, 187	vim2html.pl: 257, 259
tput: 164, 165	
tr: 183, 187	

vimdiff: 257, 259
 vimmm: 257, 259
 vimspell.sh: 257, 259
 vimtutor: 257, 259
 vipw: 180, 182
 vmstat: 174, 175
 w: 174, 175
 wall: 169, 172
 watch: 174, 175
 wc: 183, 187
 whatis: 234, 235
 whereis: 169, 172
 who: 183, 187
 whoami: 183, 187
 wipefs: 169, 172
 write: 169, 172
 write_cd_rules: 255, 256
 write_net_rules: 255, 256
 xargs: 217, 217
 xgettext: 219, 220
 xsubpp: 200, 202
 xtrace: 132, 137
 xxd: 257, 259
 xz: 232, 232
 xzcat: 232, 232
 xzdec: 232, 233
 yacc: 191, 191
 yes: 183, 187
 ylwrap: 206, 207
 zcat: 226, 226
 zcmp: 226, 226
 zdiff: 226, 226
 zdump: 132, 137
 zegrep: 226, 226
 zfgrep: 226, 226
 zforce: 226, 226
 zgrep: 226, 226
 zic: 132, 137
 zipdetails: 200, 202
 zless: 226, 226
 zmore: 226, 226
 znew: 226, 226
 zsoelim: 222, 224

libBrokenLocale: 132, 137
 libbsd-compat: 132, 137
 libbz2*: 211, 212
 libc: 132, 137
 libcidn: 132, 137
 libcloog-isl: 150, 150
 libcom_err: 177, 178
 libcrypt: 132, 137
 libcursesw: 164, 165
 libdl: 132, 137
 libe2p: 177, 178
 libee: 244, 244
 libestr: 242, 242
 libexpect-5.43: 120, 120
 libext2fs: 177, 179
 libfl.a: 195, 195, 195, 195
 libformw: 164, 165
 libg: 132, 137
 libgcc*: 156, 157
 libgcov: 156, 157
 libgettextlib: 219, 220
 libgettextpo: 219, 220
 libgettextsrc: 219, 220
 libgmp: 140, 141
 libgmpxx: 140, 141
 libgomp: 156, 157
 libhistory: 204, 204
 libiberty: 153, 155
 libieee: 132, 137
 libisl: 150, 150
 libltdl: 193, 193
 liblzma: 232, 233
 libm: 132, 137
 libmagic: 215, 215
 libmcheck: 132, 137
 libmemusage: 132, 137
 libmenuw: 164, 166
 libmount: 169, 172
 libmp: 140, 141
 libmpc: 145, 145
 libmpfr: 143, 143
 libmudflap*: 156, 157
 libncursesw: 164, 165
 libnsl: 132, 137
 libnss: 132, 137
 libopcodes: 153, 155
 libpanelw: 164, 166
 libpcprofile: 132, 137
 libppl: 147, 148
 libppl_c: 147, 148

Bibliothèques

ld.so: 132, 137
 libanl: 132, 137
 libasprintf: 219, 220
 libbfd: 153, 155
 libblkid: 169, 172

libproc: 174, 175
 libpthread: 132, 137
 libpwl: 147, 148
 libquota: 177, 179
 libreadline: 204, 204
 libresolv: 132, 137
 librpcsvc: 132, 137
 librt: 132, 137
 libSegFault: 132, 137
 libss: 177, 179
 libssp*: 156, 157
 libstdbuf: 183, 187
 libstdc++: 156, 157
 libsupc++: 156, 157
 libtcl-version.so: 119, 119
 libtclstub-version.a: 119, 119
 libthread_db: 132, 137
 libutil: 132, 137
 libuuid: 169, 172
 liby.a: 191, 191
 libz: 152, 152
 preloadable_libintl.so: 219, 220

Scripts

checkfs: 265, 265
 cleanfs: 265, 265
 console: 265, 265
 configuration: 268
 functions: 265, 265
 halt: 265, 265
 ifdown: 265, 265
 ifup: 265, 265
 localnet: 265, 265
 /etc/hosts: 278
 configuration: 278
 mountfs: 265, 265
 mountkernfs: 265, 265
 network: 265, 265
 /etc/hosts: 278
 configuration: 279
 rc: 265, 265
 reboot: 265, 265
 sendsignals: 265, 265
 setclock: 265, 265
 configuration: 268
 static: 265, 265
 swap: 265, 266
 sysklogd: 265, 266
 template: 265, 266
 udev: 265, 266

Autres

/boot/config-[linux-version]: 284, 285
 /boot/System.map-[linux-version]: 284, 285
 /dev/*: 108, 116
 /etc/clfs-release: 287
 /etc/default/grub: 261
 /etc/fstab: 106, 283
 /etc/group: 99, 114
 /etc/hosts: 278
 /etc/inittab: 95, 248
 /etc/inputrc: 276
 /etc/ld.so.conf: 135
 /etc/localtime: 134
 /etc/login.defs: 180
 /etc/nsswitch.conf: 134
 /etc/passwd: 99, 114
 /etc/profile: 274, 274
 /etc/protocols: 188
 /etc/resolv.conf: 279
 /etc/rsyslog.conf: 246
 /etc/services: 188
 /etc/udev: 255, 256
 /etc/vimrc: 258
 /lib/udev: 255, 256
 /usr/include/{asm,linux}/*.*h: 127, 127
 /var/log/btmp: 99, 114
 /var/log/lastlog: 99, 114
 /var/log/wtmp: 99, 114
 /var/run/utmp: 99, 114
 dhcpcd: 281
 man pages: 128, 128