

# **Linux From Scratch**

**Version 6.1.1**

**Gerard Beekmans**

# Linux From Scratch: Version 6.1.1

par Gerard Beekmans

Copyright © 1999–2005 Gerard Beekmans

Copyright (c) 1999–2005, Gerard Beekmans

All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions in any form must retain the above copyright notice, this list of conditions and the following disclaimer
- Neither the name of « Linux From Scratch » nor the names of its contributors may be used to endorse or promote products derived from this material without specific prior written permission
- Any material derived from Linux From Scratch must contain a reference to the « Linux From Scratch » project

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 .....	vii
1. Avant-propos .....	vii
2. Public visé .....	viii
3. Prérequis .....	x
4. Prérequis du système hôte .....	xi
5. Typographie .....	xii
6. Structure .....	xiii
7. Errata .....	xiv
I. Introduction .....	15
1. Introduction .....	16
1.1. Comment construire un système LFS .....	16
1.2. Historique des modifications .....	18
1.3. Ressources .....	20
1.4. Aide .....	21
II. Préparation de la construction .....	24
2. Préparer une nouvelle partition .....	25
2.1. Introduction .....	25
2.2. Créer une nouvelle partition .....	26
2.3. Créer un système de fichiers sur la partition .....	27
2.4. Monter la nouvelle partition .....	28
3. Paquets et correctifs .....	29
3.1. Introduction .....	29
3.2. Tous les paquets .....	30
3.3. Correctifs requis .....	34
4. Dernières préparations .....	36
4.1. À propos de \$LFS .....	36
4.2. Créer le répertoire \$LFS/tools .....	37
4.3. Ajouter l'utilisateur LFS .....	38
4.4. Configurer l'environnement .....	39
4.5. À propos des SBU .....	42
4.6. À propos des suites de tests .....	43
5. Construire un système temporaire .....	44
5.1. Introduction .....	44
5.2. Notes techniques sur l'ensemble d'outils .....	45
5.3. Binutils-2.15.94.0.2.2 - Passe 1 .....	49
5.4. GCC-3.4.3 - Passe 1 .....	51
5.5. Linux-Libc-Headers-2.6.11.2 .....	53
5.6. Glibc-2.3.4 .....	54
5.7. Ajuster l'atelier des outils .....	57
5.8. Tcl-8.4.9 .....	59
5.9. Expect-5.43.0 .....	61
5.10. DejaGNU-1.4.4 .....	63
5.11. GCC-3.4.3 - Passe 2 .....	64
5.12. Binutils-2.15.94.0.2.2 - Passe 2 .....	67
5.13. Gawk-3.1.4 .....	69
5.14. Coreutils-5.2.1 .....	70
5.15. Bzip2-1.0.3 .....	71

5.16. Gzip-1.3.5 .....	72
5.17. Diffutils-2.8.1 .....	73
5.18. Findutils-4.2.23 .....	74
5.19. Make-3.80 .....	75
5.20. Grep-2.5.1a .....	76
5.21. Sed-4.1.4 .....	77
5.22. Gettext-0.14.3 .....	78
5.23. Ncurses-5.4 .....	79
5.24. Patch-2.5.4 .....	80
5.25. Tar-1.15.1 .....	81
5.26. Texinfo-4.8 .....	82
5.27. Bash-3.0 .....	83
5.28. M4-1.4.3 .....	84
5.29. Bison-2.0 .....	85
5.30. Flex-2.5.31 .....	86
5.31. Util-linux-2.12q .....	87
5.32. Perl-5.8.7 .....	88
5.33. Supprimer les symboles des fichiers objets .....	89
III. Construction du système LFS .....	90
6. Installer les logiciels du système de base .....	91
6.1. Introduction .....	91
6.2. Monter les systèmes de fichiers virtuels du noyau .....	92
6.3. Entrer dans l'environnement chroot .....	93
6.4. Changer de propriétaire .....	94
6.5. Créer les répertoires .....	95
6.6. Créer les liens symboliques essentiels .....	96
6.7. Créer les fichiers passwd, group et les journaux de trace .....	97
6.8. Peupler /dev .....	99
6.9. Linux-Libc-Headers-2.6.11.2 .....	101
6.10. Man-pages-2.01 .....	102
6.11. Glibc-2.3.4 .....	103
6.12. Ré-ajustement de l'ensemble d'outils .....	110
6.13. Binutils-2.15.94.0.2.2 .....	112
6.14. GCC-3.4.3 .....	115
6.15. Coreutils-5.2.1 .....	118
6.16. Zlib-1.2.3 .....	123
6.17. Mktmp-1.5 .....	125
6.18. Iana-Etc-1.04 .....	126
6.19. Findutils-4.2.23 .....	127
6.20. Gawk-3.1.4 .....	129
6.21. Ncurses-5.4 .....	130
6.22. Readline-5.0 .....	132
6.23. Vim-6.3 .....	134
6.24. M4-1.4.3 .....	137
6.25. Bison-2.0 .....	138
6.26. Less-382 .....	139
6.27. Groff-1.19.1 .....	140
6.28. Sed-4.1.4 .....	143
6.29. Flex-2.5.31 .....	144
6.30. Gettext-0.14.3 .....	146
6.31. Inetutils-1.4.2 .....	148
6.32. IPRoute2-2.6.11-050330 .....	150

6.33. Perl-5.8.7 .....	152
6.34. Texinfo-4.8 .....	154
6.35. Autoconf-2.59 .....	156
6.36. Automake-1.9.5 .....	158
6.37. Bash-3.0 .....	160
6.38. File-4.13 .....	162
6.39. Libtool-1.5.14 .....	163
6.40. Bzip2-1.0.3 .....	164
6.41. Diffutils-2.8.1 .....	166
6.42. Kbd-1.12 .....	167
6.43. E2fsprogs-1.37 .....	169
6.44. Grep-2.5.1a .....	172
6.45. Grub-0.96 .....	173
6.46. Gzip-1.3.5 .....	175
6.47. Hotplug-2004_09_23 .....	177
6.48. Man-1.5p .....	179
6.49. Make-3.80 .....	181
6.50. Module-Init-Tools-3.1 .....	182
6.51. Patch-2.5.4 .....	184
6.52. Procps-3.2.5 .....	185
6.53. Psmisc-21.6 .....	187
6.54. Shadow-4.0.9 .....	188
6.55. Sysklogd-1.4.1 .....	192
6.56. Sysvinit-2.86 .....	194
6.57. Tar-1.15.1 .....	197
6.58. Udev-056 .....	198
6.59. Util-linux-2.12q .....	200
6.60. Symboles de débogage .....	204
6.61. Supprimer de nouveau les symboles des fichiers objets .....	205
6.62. Nettoyer .....	206
7. Initialiser les scripts de démarrage du système .....	207
7.1. Introduction .....	207
7.2. LFS-Bootscripts-3.2.1 .....	208
7.3. Fonctionnement des scripts de démarrage .....	210
7.4. Gestion des périphériques et modules sur un système LFS .....	212
7.5. Configurer le script setclock .....	216
7.6. Configurer la console Linux .....	217
7.7. Configurer le script sysklogd .....	219
7.8. Créer le fichier /etc/inputrc .....	220
7.9. Les fichiers de démarrage du shell Bash .....	222
7.10. Configurer le script localnet .....	225
7.11. Créer le fichier /etc/hosts .....	226
7.12. Configurer le script network .....	227
8. Rendre le système LFS amorçable .....	229
8.1. Introduction .....	229
8.2. Créer le fichier /etc/fstab .....	230
8.3. Linux-2.6.11.12 .....	231
8.4. Rendre le système LFS amorçable .....	234
9. Fin .....	236
9.1. La fin .....	236
9.2. Enregistrez-vous .....	237
9.3. Redémarrer le système .....	238

9.4. Et maintenant? .....	239
IV. Annexes .....	240
A. Acronymes et termes .....	241
B. Remerciements .....	244
Index .....	248

# Préface

## 1. Avant-propos

Mes aventures dans Linux ont commencé en 1998 lorsque j'ai téléchargé et installé ma première distribution. Après avoir travaillé dessus un bon moment, j'ai découvert des problèmes que j'aurais vraiment aimé voir améliorer. Par exemple, je n'aimais pas l'arrangement des scripts de démarrage ou la façon dont les programmes étaient configurés par défaut. J'ai essayé un certain nombre d'autres distributions pour corriger ces problèmes, cependant chacune avait ses avantages et ses inconvénients. Finalement, j'ai réalisé que si je voulais avoir une pleine satisfaction de mon système Linux, je devais le construire à partir de rien.

Qu'est-ce que cela signifie ? Je me suis résolu à ne pas utiliser de paquets déjà compilés, quels qu'ils soient, et à ne pas utiliser de CD-ROM ou de disques d'amorçage qui installeraient des outils de base. J'utiliserais mon système Linux actuel pour développer mon propre système personnalisé. Ce système Linux « parfait » aurait alors la force des autres systèmes sans avoir leurs faiblesses. Au début, l'idée était un peu écrasante mais j'ai conservé l'idée qu'un système pourrait être construit en se conformant à mes besoins et désirs plutôt qu'à un standard qui ne correspondrait pas à ce que je cherchais.

Après avoir passé quelques problèmes comme des dépendances circulaires et erreurs à la compilation, j'ai créé un système Linux personnalisé entièrement opérationnel et convenant à des besoins individuels. Ce processus m'a aussi permis de créer des systèmes Linux compacts et précis, bien plus rapides et prenant moins de place que des systèmes d'exploitation traditionnels. J'ai appelé ce système LFS, soit « Linux From Scratch » (Linux à partir de rien).

Lorsque j'ai partagé mes objectifs et mes expériences avec d'autres membres de la communauté Linux, il est devenu apparent qu'il y avait un sérieux intérêt dans les idées que j'avais mises en avant lors de mes aventures Linux. De tels systèmes LFS personnalisés rencontraient non seulement les spécifications et pré-requis des utilisateurs mais servaient aussi comme opportunité idéale d'apprentissage pour les programmeurs et les administrateurs système, afin d'améliorer leurs connaissances sous Linux. De cet intérêt est né le projet « Linux From Scratch ».

Le livre « *Linux From Scratch* » fournit aux lecteurs la base et les instructions pour concevoir et créer des systèmes Linux personnalisés. Ce livre met en lumière le projet Linux from Scratch et les bénéfices de l'utilisation de ce système. Les utilisateurs peuvent dicter tous les aspects de leur système, ceci incluant la répartition des répertoires, la configuration des scripts et la sécurité. Le système résultant sera compilé directement à partir du code source et l'utilisateur sera capable de spécifier où, pourquoi et comment les programmes sont installés. Ce livre permet aux lecteurs de personnaliser complètement les systèmes Linux suivant leurs besoins et donne plus de contrôle aux utilisateurs sur leur système.

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

--

Gerard Beekmans  
gerard@linuxfromscratch.org

## 2. 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 LFS à partir du code source. La question que beaucoup de personnes se posent est « pourquoi se fatiguer à installer manuellement un système Linux depuis le début alors qu'il suffit de télécharger une distribution existante ? ». C'est une bonne question.

Une raison importante de l'existence de LFS est d'apprendre comment fonctionne un système Linux de l'intérieur. Construire un système LFS 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 LFS est qu'il permet à l'utilisateur d'avoir plus de contrôle sur son système sans avoir à dépendre d'une implémentation créée par quelqu'un d'autre. Avec LFS, *vous* êtes maintenant sur le siège du conducteur et *vous* êtes capable de décider chaque aspect de votre 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 LFS est la possibilité de créer un système Linux très compact. Lors de l'installation d'une distribution habituelle, l'utilisateur finit par inclure beaucoup de programmes qui ne seront jamais utilisés. Ces programmes occupent de l'espace disque et font parfois perdre des cycles CPU précieux. Il n'est pas difficile de construire un système LFS 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 système LFS minuscule. Nous avons installé un système spécialisé pour faire fonctionner le serveur web Apache ; l'espace disque total occupé était approximativement de 8 Mo. 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 seulement 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 au restaurant fast-food, vous n'avez aucune idée de ce que vous mangez. LFS ne vous donne pas un hamburger, mais la recette pour faire un hamburger. Cela permet aux utilisateurs de prudemment l'inspecter, d'enlever les ingrédients non désirés et, par la même occasion, de rajouter des ingrédients qui correspondent mieux à la saveur qu'ils attendent de ce hamburger. Quand vous êtes satisfait des ingrédients, 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 LFS à une maison construite. LFS fournit les plans de la maison, mais c'est à vous de la construire. LFS 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. Malgré tout vous n'avez aucune garantie que le nouveau paquet résolve le problème de manière adéquate. Vous ne pourrez jamais savoir si une faille de sécurité est réparée si vous ne le faites pas vous-même.

Le but de 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, sur le site du projet de documentation Linux (TLDP) <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 il 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 LFS pour pouvoir toutes les lister ici. Cette section n'aborde que la partie visible de l'iceberg. En continuant dans votre expérience de LFS, vous trouverez la puissance que donnent l'information et la connaissance.

### 3. Prérequis

Construire un système LFS n'est pas une tâche simple. Cela requiert un certain niveau de connaissance en administration de système Unix pour résoudre les problèmes et exécuter correctement les commandes listées. En particulier, comme barrière 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 répertoire et de fichier, et pour changer de répertoire. Il est aussi attendu que le lecteur dispose d'une connaissance raisonnable dans l'utilisation et l'installation de logiciels Linux.

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

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

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

C'est un guide complet sur la construction et l'installation « générique » de logiciels Unix sous Linux.

- 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://www.linuxfromscratch.org/hints/downloads/files/essential\\_prereading.txt](http://www.linuxfromscratch.org/hints/downloads/files/essential_prereading.txt) (NdT : Astuce essentielle à lire avant tout)

C'est une astuce LFS écrite spécifiquement pour les nouveaux utilisateurs Linux. C'est principalement une liste de liens vers des 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...

## 4. Prérequis du système hôte

Le noyau installé sur l'hôte doit être au minimum un noyau 2.6.2 compilé avec GCC-3.0 ou ultérieur. Deux raisons expliquent ce prérequis. Tout d'abord, la suite de tests NPTL (*Native POSIX Threading Library*, soit la bibliothèque des threads POSIX) causera une erreur de segmentation si le noyau de l'hôte n'a pas été compilé avec GCC-3.0 ou avec une version ultérieure. Ensuite, la version 2.6.2 ou ultérieure du noyau est nécessaire pour utiliser Udev. Udev crée les périphériques dynamiquement en les découvrant dans le système de fichiers `sysfs`. Néanmoins, le support de ce système de fichier a été récemment implémenté dans la plupart des pilotes du noyau. Nous voulons être sûr que tous les périphériques système critiques seront proprement créés.

Pour déterminer si le noyau de l'hôte valide les prérequis indiqués ci-dessus, lancez la commande suivante :

```
cat /proc/version
```

Elle devrait produire un affichage similaire à :

```
Linux version 2.6.2 (user@host) (gcc version 3.4.0) #1  
Tue Apr 20 21:22:18 GMT 2004
```

Si le résultat de la commande ci-dessus indique que la version du noyau de l'hôte est soit 2.6.2 (ou ultérieur) ou qu'il n'a pas été compilé en utilisant GCC-3.0 (ou ultérieur), vous aurez besoin de l'installer. Il existe deux méthodes pour résoudre ceci. Tout d'abord, voir si votre distribution Linux fournit un noyau 2.6.2 (ou ultérieur). Dans ce cas, vous pouvez l'installer. Si votre distribution ne le propose pas ou si vous préférez ne pas l'installer, alors vous pouvez compiler un noyau 2.6 vous-même. Les instructions de compilation d'un noyau et de configuration du chargeur de démarrage (en supposant que l'hôte utilise GRUB) sont situés dans Chapitre 8. Cette deuxième option peut aussi être vue comme un test de vos connaissances Linux. Si cette étape est trop importante, alors le livre LFS ne vous sera pas d'une grande utilité actuellement.

## 5. Typographie

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

```
./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`.

### *Emphase*

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

<http://www.linuxfromscratch.org/>

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

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

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

[TEXTE À REMPLACER]

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

`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 situés 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 programme]** est généralement suffisant.

## 6. Structure

Ce livre est divisé en plusieurs parties.

### 6.1. Partie I - Introduction

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

### 6.2. Partie II - Préparation de la construction

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

### 6.3. Partie III - Construction du système LFS

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

## 7. Errata

Le logiciel utilisé pour créer un système LFS est constamment mis à jour et amélioré. Les messages d'avertissements pour la sécurité et les corrections de bogues pourraient survenir après la sortie du livre LFS. Pour vérifier si les versions du paquetage ou les instructions de cette version de LFS ont besoin de modifications pour corriger les vulnérabilités en terme de sécurité ou toute autre correction de bogue, merci de visiter <http://www.linuxfromscratch.org/lfs/errata/6.1.1/> avant de commencer votre construction. Vous devez noter toute modification et l'appliquer à la bonne section du livre pendant votre progression lors de la construction du système LFS.

# Introduction

# Chapitre 1. Introduction

## 1.1. Comment construire un système LFS

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

Comme alternative à l'installation d'une distribution séparée complète sur votre machine, vous pouvez utiliser le LiveCD Linux From Scratch. Le CD fonctionne bien en tant que système hôte, fournissant tous les outils dont vous avez besoin pour suivre les instructions de ce livre avec succès. De plus, il contient les paquetages sources, correctifs et une copie de ce livre. Une fois que vous avez le CD, aucune connexion réseau et aucun téléchargement supplémentaire n'est nécessaire. Pour plus d'informations sur le LiveCD LFS ou pour télécharger une copie, visitez <http://www.linuxfromscratch.org/livecd/>.

Le Chapitre 2 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 LFS sera compilé et installé. Le Chapitre 3 explique quels paquets et correctifs ont besoin d'être téléchargés pour construire un système LFS et comment les stocker sur le nouveau système de fichiers. Le Chapitre 4 discute de la configuration pour un environnement de travail approprié. Merci de lire le Chapitre 4 avec attention car il explique plusieurs problèmes importants dont le développeur doit être conscient avant de commencer à travailler sur le Chapitre 5 et les chapitres suivants.

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

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

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

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

Pour terminer l'installation, les scripts de démarrage sont configurés dans le Chapitre 7, le noyau et le chargeur de démarrage sont configurés dans le Chapitre 8. Le Chapitre 9 contient des informations sur la



suite de l'expérience LFS après ce livre. Après l'implémentation des étapes de ce livre, l'ordinateur sera prêt à redémarrer dans le nouveau système LFS.

Voilà le processus en quelques mots. Des informations détaillées sur chaque étape sont fournies dans les chapitres suivants avec les descriptions des paquets. Les éléments qui peuvent sembler compliqués seront clarifiés et tout se mettra en place à mesure que le lecteur s'embarquera pour l'aventure LFS.

## 1.2. Historique des modifications

Il s'agit de la version 6.1.1 du livre « Linux From Scratch », datant du 30 novembre 2005. Si ce livre est daté de plus de six mois, une nouvelle et meilleure version est probablement déjà disponible. Pour le savoir, merci de vérifier la présence d'une nouvelle version sur l'un des miroirs via <http://www.linuxfromscratch.org/>.

Ci-dessous se trouve une liste des modifications apportées depuis la version précédente du livre avec un résumé suivi d'une explication plus détaillée.

- Paquets mis à jour :
  - Perl 5.8.7
  - Zlib 1.2.3
- Paquets ajoutés :
  - binutils-2.15.94.0.2.2-gcc4-1.patch
  - bzip2-1.0.3-install\_docs-1.patch
  - bzip2-1.0.3-bzgrep\_security-1.patch
  - glibc-2.3.4-rtld\_search\_dirs-1.patch
  - glibc-2.3.4-tls\_assert-1.patch
  - texinfo-4.8-tempfile\_fix-1.patch
  - util-linux-2.12q-umount\_fix-1.patch
  - vim-6.3-security\_fix-2.patch
- Paquets supprimés :
  - zlib-1.2.2-security\_fix-1.patch;
- 30 novembre 2005 [matt] : sortie de LFS-6.1.1.
- 24 novembre 2005 [matt] : sortie de LFS-6.1.1-pre2.
- 24 novembre 2005 [matt] : correction d'un problème avec Glibc empêchant certains programmes (comme OpenOffice.org) de fonctionner.
- 23 novembre 2005 [gerard] : correction d'une référence aux 'pages man' par la 'documentation HTML' dans le chapitre 6/sec
- 18 novembre 2005 [manuel] : correction du déballage du paquet module-init-tools-testsuite.
- 18 novembre 2005 [manuel] : corrections du PDF.
- 17 novembre 2005 [matt] : sortie de LFS-6.1.1-pre1.
- 12 novembre 2005 [matt] : amélioration de l'heuristique déterminant une locale supportée à la fois par Glibc et les paquets en dehors de LFS (bogue 1642). Merci à Alexander Patrakov pour avoir mis en évidence les nombreux problèmes et pour avoir donné son avis sur les nombreux et divers correctifs suggérés.

- 12 novembre 2005 [matt] : omission de l'exécution de la suite de tests de Bzip2 en tant qu'étape séparée car **make** l'exécute automatiquement (bogue 1652).
- 7 novembre 2005 [matt] : empêche Udev de tuer les processus udevd du système hôte (bogue 1651). Merci à Alexander Patrakov pour l'information et la correction.
- 5 novembre 2005 [matt] : ajout d'une note sur la vérification de la propreté des outils de construction dans le chapitre 5 en expliquant que, si TCL échoue lors de sa construction, c'est un signe qu'un outil de construction est cassé (bogue 1581).
- 4 novembre 2005 [matt] : corrections des instructions sur l'exécution de la suite de tests Module-Init-Tools (bogue 1597). Merci à Greg Schafer, Tushar Teredesai et Randy McMurphy pour le correctif fourni.
- 29 octobre 2005 [manuel] : corrections du PDF.
- 23 octobre 2005 [manuel] : ajout de l'installation de la documentation de Bash. Ajout de notes sur libiconv et Cracklib. Correction de l'installation de la documentation de Sed. Remplacement d'un correctif pour IPRoute2 par une commande sed.
- 19 octobre 2005 [manuel] : mise à jour des remerciements avec la version actuelle du trunk. Portage des modifications de rédaction dans la préface et le chapitre 1. Déplacement du chapitre 2 dans la partie II. Ajout des options -v. Portage de quelques corrections de typo et de rédaction provenant du trunk.
- 19 octobre 2005 [manuel] : mise à jour des feuilles de style, du Makefile et des fichiers relatifs par les versions du trunk.
- 15 octobre 2005 [matt] : utilisation d'une version mise à jour du fichier de règles d'Udev (bogue 1639).
- 15 octobre 2005 [matt] : ajout d'un groupe cdrom, requis par le fichier de règles d'Udev.
- 14 octobre 2005 [ken] : ajout d'un correctif permettant à binutils d'être construit sur un hôte fonctionnant avec gcc-4, mise à jour des instructions de glibc pour le correctif rtdl, mise à jour de space/time pour perl et zlib.
- 14 octobre 2005 [matt] : ajout d'un correctif pour gérer la faille de sécurité dans util-linux.
- 14 octobre 2005 [matt] : ajout d'un correctif de sécurité mis à jour pour vim.
- 14 octobre 2005 [jhuntwork] : ajout de correctifs pour la sécurité et les documents d'installation de bzip2.
- 14 octobre 2005 [jhuntwork] : ajout d'un correctif sur tempfile pour texinfo.
- 14 octobre 2005 [ken] : mise à jour de paquets et de correctifs dans le journal des modifications pour simplement refléter les changements depuis 6.1. Mise à jour de zlib.
- 13 octobre 2005 [ken] : correction d'erreurs connues dans les listes de fichiers installés et augmentation de la version de perl.

## 1.3. Ressources

### 1.3.1. FAQ

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

### 1.3.2. Listes de diffusion

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

Pour connaître les listes disponibles, les conditions d'abonnement, l'emplacement des archives et quelques autres informations, allez sur <http://www.linuxfromscratch.org/mail.html>.

### 1.3.3. Serveur de nouvelles

Les listes de diffusion gérées par `linuxfromscratch.org` sont aussi accessibles via le serveur NNTP. Tous les messages envoyés sur une liste de diffusion sont copiés dans le groupe de nouvelles correspondant, et vice-versa.

Le serveur de nouvelles est situé sur [news.linuxfromscratch.org](http://news.linuxfromscratch.org).

### 1.3.4. IRC

Plusieurs membres de la communauté LFS offrent une assistance sur le réseau IRC (Internet Relay Chat) de notre communauté. Avant d'utiliser ce mode de support, assurez-vous que la réponse à votre question ne se trouve pas déjà dans la FAQ LFS (voir ci-dessus) ou dans les archives des listes de diffusion (voir ci-dessous) pour tenter de trouver une réponse à votre question. Vous trouverez le réseau IRC à l'adresse `irc.linuxfromscratch.org`, port 6667. Le canal du support se nomme `#LFS-support`.

### 1.3.5. Références

Pour plus d'informations sur les paquets, des indications très utiles sont disponibles dans la page de référence des paquetages LFS située sur : <http://www.109bean.org.uk/LFS-references.html>.

### 1.3.6. Sites miroirs

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

### 1.3.7. Contacts

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

## 1.4. Aide

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

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

Nous avons aussi une formidable communauté LFS, volontaire pour offrir une assistance via les listes de discussion et IRC (voir la section Section 1.3, « 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.

### 1.4.1. 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, 6.1.1)
- La distribution hôte (et sa version) que vous utilisez pour créer LFS
- Le paquet ou la section où le problème a été rencontré
- Le message d'erreur exact ou le symptôme reçu
- Notez si vous avez dévié du livre



#### Note

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

### 1.4.2. 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.

### 1.4.3. Problèmes de compilation

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

```

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

```

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

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

Cette information n'est pas suffisante pour diagnostiquer correctement le problème car cela montre 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 sauvé 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.



# Préparation de la construction



## Chapitre 2. Préparer une nouvelle partition

### 2.1. Introduction

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

## 2.2. Créer une nouvelle partition

Comme la plupart des autres systèmes d'exploitation, LFS est habituellement installé dans une partition dédiée. L'approche recommandée pour la construction d'un système LFS est d'utiliser une partition vide disponible ou, si vous avez assez d'espace non partitionné, d'en créer une. Néanmoins, un système LFS (en fait, même plusieurs systèmes LFS) peuvent aussi être installés sur une partition déjà occupée par un autre système d'exploitation. Les différents systèmes cohabiteront en paix. Le document [http://www.linuxfromscratch.org/hints/downloads/files/lfs\\_next\\_to\\_existing\\_systems.txt](http://www.linuxfromscratch.org/hints/downloads/files/lfs_next_to_existing_systems.txt) explique comment implémenter ceci alors que ce livre se base sur la méthode utilisant une partition vierge pour l'installation.

Un système minimal requiert une partition d'environ 1,3 Go. C'est suffisant pour conserver toutes les archives tar des sources et pour compiler tous les paquets. Néanmoins, si le système LFS a pour but d'être un système Linux primaire, des logiciels supplémentaires seront probablement installés et réclameront une place supplémentaire (entre 2 et 3 Go). Le système LFS lui-même ne prendra pas tout cet espace. Une grande partie de cet espace est requis pour fournir un espace libre suffisant mais temporaire. Compiler des paquets peut demander beaucoup d'espace disque qui sera récupéré après l'installation du paquet.

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

Lancez un programme de partitionnement de disques tel que **cdisk** 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 IDE. Créez une partition Linux native et, si nécessaire, une partition de swap. Merci de vous référer aux pages man de **cdisk(8)** ou **fdisk(8)** si vous ne savez pas encore utiliser le programme.

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

## 2.3. Créer un système de fichiers sur la partition

Maintenant qu'une partition vierge est prête, le système de fichiers peut être créé. Le système le plus communément utilisé dans le monde Linux est le système de fichiers étendu, deuxième version, (plus connu sous son acronyme, ext2), mais avec les nouveaux disques haute capacité, les systèmes de fichiers journalisés deviennent de plus en plus populaires. Nous créerons un système de fichiers ext2. Les instructions de construction d'autres systèmes de fichiers sont disponibles dans <http://www.linuxfromscratch.org/blfs/view/svn/postlfs/filesystems.html>.

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

```
mke2fs -v /dev/[xxx]
```

Remplacez `[xxx]` avec le nom de la partition LFS (`hda5` dans notre exemple précédent).



### Note

Quelques distributions hôtes utilisent des fonctionnalités personnalisées dans leur outil de création de systèmes de fichiers (e2fsprogs). Ceci peut poser des problèmes lors du démarrage dans votre nouveau LFS au chapitre 9 car toutes ces fonctionnalités ne seront pas supportées par la version d'e2fsprogs installée par LFS si vous obtenez une erreur similaire à « unsupported filesystem features, upgrade your e2fsprogs ». Pour vérifier que 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 paquetage e2fsprogs et utiliser les binaires résultant de cette compilation pour re-créeer le système de fichiers sur votre partition LFS :

```
cd /tmp
tar -xjvf /path/to/sources/e2fsprogs-1.37.tar.bz2
cd e2fsprogs-1.37
mkdir -v build
cd build
../configure
make # notez que nous n'exécutons pas 'make install' ici
      # de façon intentionnelle !
./misc/mke2fs -v /dev/[xxx]
cd /tmp
rm -rfv e2fsprogs-1.37
```

Si une partition de swap a été créée, elle devra être initialisée, pour pouvoir être utilisée, en exécutant la commande ci-dessous. Si vous utilisez déjà une partition de swap, il n'est pas nécessaire de la formater.

```
mkswap -v /dev/[yyy]
```

Remplacez `[yyy]` avec le nom de la partition de swap.

## 2.4. Monter la nouvelle partition

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

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

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

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

```
mkdir -pv $LFS
mount /dev/[xxx] $LFS
```

Remplacez `[xxx]` avec la désignation de la partition LFS.

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

```
mkdir -pv $LFS
mount -v /dev/[xxx] $LFS
mkdir -v $LFS/usr
mount -v /dev/[yyy] $LFS/usr
```

Remplacez `[xxx]` et `[yyy]` avec 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 configurées pour la partition LFS 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 ces 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. Le 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.

Les emplacements de téléchargements pourraient ne pas être toujours accessibles. Si un emplacement de téléchargement a changé depuis la publication de ce livre, Google (<http://www.google.com/>) fournit un outil de recherche puissant pour la plupart des paquets. Si cette recherche n'a pas de succès, essayez un autre moyen de téléchargement parmi ceux qui sont disponibles sur <http://www.linuxfromscratch.org/lfs/packages.html>.

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

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

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

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

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

## 3.2. Tous les paquets

Téléchargez ou obtenez autrement les paquets suivants :

- Autoconf (2.59) - 908 Ko :  
*<http://ftp.gnu.org/gnu/autoconf/>*
- Automake (1.9.5) - 748 Ko :  
*<http://ftp.gnu.org/gnu/automake/>*
- Bash (3.0) - 1 824 Ko :  
*<http://ftp.gnu.org/gnu/bash/>*
- Bash Documentation (3.0) - 1 994 Ko :  
*<http://ftp.gnu.org/gnu/bash/>*
- Binutils (2.15.94.0.2.2) - 11 056 Ko :  
*<http://www.kernel.org/pub/linux/devel/binutils/>*
- Bison (2.0) - 916 Ko :  
*<http://ftp.gnu.org/gnu/bison/>*
- Bzip2 (1.0.3) - 596 Ko :  
*<http://www.bzip.org/>*
- Coreutils (5.2.1) - 4 184 Ko :  
*<http://ftp.gnu.org/gnu/coreutils/>*
- DejaGNU (1.4.4) - 852 Ko :  
*<http://ftp.gnu.org/gnu/dejagnu/>*
- Diffutils (2.8.1) - 648 Ko :  
*<http://ftp.gnu.org/gnu/diffutils/>*
- E2fsprogs (1.37) - 3 100 Ko :  
*<http://prdownloads.sourceforge.net/e2fsprogs/>*
- Expect (5.43.0) - 416 Ko :  
*<http://expect.nist.gov/src/>*
- File (4.13) - 324 Ko :  
*<ftp://ftp.gw.com/mirrors/pub/unix/file/>*



### Note

File (4.13) pourrait ne plus être disponible à l'emplacement indiqué. Les administrateurs du site de téléchargement principal suppriment de temps en temps les anciennes versions lorsque de nouvelles sont disponibles. Un autre emplacement de téléchargement, pouvant avoir à disposition la bonne version, se trouve aussi sur <http://www.linuxfromscratch.org/lfs/download.html#ftp>.

- Findutils (4.2.23) - 784 Ko :  
*<http://ftp.gnu.org/gnu/findutils/>*
- Flex (2.5.31) - 672 Ko :  
*<http://prdownloads.sourceforge.net/lex/>*

- Gawk (3.1.4) - 1 696 Ko :  
<http://ftp.gnu.org/gnu/gawk/>
- GCC (3.4.3) - 26 816 Ko :  
<http://ftp.gnu.org/gnu/gcc/>
- Gettext (0.14.3) - 4 568 Ko :  
<http://ftp.gnu.org/gnu/gettext/>
- Glibc (2.3.4) - 12 924 Ko :  
<http://ftp.gnu.org/gnu/glibc/>
- Glibc-Linuxthreads (2.3.4) - 236 Ko :  
<http://ftp.gnu.org/gnu/glibc/>
- Grep (2.5.1a) - 520 Ko :  
<http://ftp.gnu.org/gnu/grep/>
- Groff (1.19.1) - 2 096 Ko :  
<http://ftp.gnu.org/gnu/groff/>
- GRUB (0.96) - 768 Ko :  
<ftp://alpha.gnu.org/gnu/grub/>
- Gzip (1.3.5) - 284 Ko :  
<ftp://alpha.gnu.org/gnu/gzip/>
- Hotplug (2004\_09\_23) - 40 Ko :  
<http://www.kernel.org/pub/linux/utils/kernel/hotplug/>
- Iana-Etc (1.04) - 176 Ko :  
<http://www.sethwicklein.net/projects/iana-etc/downloads/>
- Inetutils (1.4.2) - 752 Ko :  
<http://ftp.gnu.org/gnu/inetutils/>
- IPRoute2 (2.6.11-050330) - 276 Ko :  
<http://developer.osdl.org/dev/iproute2/download/>
- Kbd (1.12) - 624 Ko :  
<http://www.kernel.org/pub/linux/utils/kbd/>
- Less (382) - 216 Ko :  
<http://ftp.gnu.org/gnu/less/>
- LFS-Bootscripts (3.2.1) - 32 Ko :  
<http://downloads.linuxfromscratch.org/>
- Libtool (1.5.14) - 1 604 Ko :  
<http://ftp.gnu.org/gnu/libtool/>
- Linux (2.6.11.12) - 35 792 Ko :  
<http://www.kernel.org/pub/linux/kernel/v2.6/>
- Linux-Libc-Headers (2.6.11.2) - 2 476 Ko :  
<http://ep09.pld-linux.org/~mmazur/linux-libc-headers/>
- M4 (1.4.3) - 304 Ko :  
<http://ftp.gnu.org/gnu/m4/>
- Make (3.80) - 904 Ko :

<http://ftp.gnu.org/gnu/make/>

- Man (1.5p) - 208 Ko :  
<http://www.kernel.org/pub/linux/utils/man/>
- Man-pages (2.01) - 1 640 Ko :  
<http://www.kernel.org/pub/linux/docs/manpages/>
- Mktmp (1.5) - 68 Ko :  
<ftp://ftp.mktemp.org/pub/mktemp/>
- Module-Init-Tools (3.1) - 128 Ko :  
<http://www.kernel.org/pub/linux/utils/kernel/module-init-tools/>
- Module-Init-Tools-Testsuite (3.1) - 34 Ko :  
<http://www.kernel.org/pub/linux/utils/kernel/module-init-tools/>
- Ncurses (5.4) - 1 556 Ko :  
<ftp://invisible-island.net/ncurses/>
- Patch (2.5.4) - 156 Ko :  
<http://ftp.gnu.org/gnu/patch/>
- Perl (5.8.7) - 9 628 Ko :  
<http://ftp.funet.fi/pub/CPAN/src/>
- Procps (3.2.5) - 224 Ko :  
<http://procps.sourceforge.net/>
- Psmisc (21.6) - 188 Ko :  
<http://prdownloads.sourceforge.net/psmisc/>
- Readline (5.0) - 1 456 Ko :  
<http://ftp.gnu.org/gnu/readline/>
- Sed (4.1.4) - 632 Ko :  
<http://ftp.gnu.org/gnu/sed/>
- Shadow (4.0.9) - 1 084 Ko :  
<ftp://ftp.pld.org.pl/software/shadow/>



### Note

Shadow (4.0.9) pourrait ne plus être disponible à l'emplacement indiqué. Les administrateurs du site de téléchargement principal suppriment de temps en temps les anciennes versions lorsque de nouvelles sont disponibles. Un autre emplacement de téléchargement, pouvant avoir à disposition la bonne version, est  
<http://www.linuxfromscratch.org/lfs/download.html#ftp>.

- Sysklogd (1.4.1) - 72 Ko :  
<http://www.infodrom.org/projects/sysklogd/download/>
- Sysvinit (2.86) - 88 Ko :  
<ftp://ftp.cistron.nl/pub/people/miquels/sysvinit/>
- Tar (1.15.1) - 1 580 Ko :  
<http://ftp.gnu.org/gnu/tar/>
- Tcl (8.4.9) - 2 748 Ko :  
<http://prdownloads.sourceforge.net/tcl/>



- Texinfo (4.8) - 1 492 Ko :  
<http://ftp.gnu.org/gnu/texinfo/>
- Udev (056) - 476 Ko :  
<http://www.kernel.org/pub/linux/utils/kernel/hotplug/>
- Udev Rules Configuration - 5 Ko :  
<http://downloads.linuxfromscratch.org/udev-config-4.rules>
- Util-linux (2.12q) - 1 344 Ko :  
<http://www.kernel.org/pub/linux/utils/util-linux/>
- Vim (6.3) - 3 620 Ko :  
<ftp://ftp.vim.org/pub/vim/unix/>
- Fichiers de langue Vim (6.3) (optionnel) - 540 Ko :  
<ftp://ftp.vim.org/pub/vim/extra/>
- Zlib (1.2.3) - 415 Ko :  
<http://www.zlib.net/>

Taille totale de ces paquets : 146 Mo

### 3.3. Correctifs requis

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

- Bash Avoid Wcontinued Patch - 1 Ko :  
*[http://www.linuxfromscratch.org/patches/lfs/6.1.1/bash-3.0-avoid\\_WCONTINUED-1.patch](http://www.linuxfromscratch.org/patches/lfs/6.1.1/bash-3.0-avoid_WCONTINUED-1.patch)*
- Bash Various Fixes - 23 Ko :  
*<http://www.linuxfromscratch.org/patches/lfs/6.1.1/bash-3.0-fixes-3.patch>*
- Binutils Build From Host Running Gcc4 Patch - 2 Ko :  
*<http://www.linuxfromscratch.org/patches/lfs/6.1.1/binutils-2.15.94.0.2.2-gcc4-1.patch>*
- Bzip2 Documentation Patch - 1 Ko :  
*[http://www.linuxfromscratch.org/patches/lfs/6.1.1/bzip2-1.0.3-install\\_docs-1.patch](http://www.linuxfromscratch.org/patches/lfs/6.1.1/bzip2-1.0.3-install_docs-1.patch)*
- Bzip2 Bzgrep Security Fixes Patch - 1 Ko :  
*[http://www.linuxfromscratch.org/patches/lfs/6.1.1/bzip2-1.0.3-bzgrep\\_security-1.patch](http://www.linuxfromscratch.org/patches/lfs/6.1.1/bzip2-1.0.3-bzgrep_security-1.patch)*
- Coreutils Suppress Uptime, Kill, Su Patch - 15 Ko :  
*[http://www.linuxfromscratch.org/patches/lfs/6.1.1/coreutils-5.2.1-suppress\\_uptime\\_kill\\_su-1.patch](http://www.linuxfromscratch.org/patches/lfs/6.1.1/coreutils-5.2.1-suppress_uptime_kill_su-1.patch)*
- Coreutils Uname Patch - 4 Ko :  
*<http://www.linuxfromscratch.org/patches/lfs/6.1.1/coreutils-5.2.1-uname-2.patch>*
- Expect Spawn Patch - 7 Ko :  
*<http://www.linuxfromscratch.org/patches/lfs/6.1.1/expect-5.43.0-spawn-1.patch>*
- Flex Brokenness Patch - 156 Ko :  
*[http://www.linuxfromscratch.org/patches/lfs/6.1.1/flex-2.5.31-debian\\_fixes-3.patch](http://www.linuxfromscratch.org/patches/lfs/6.1.1/flex-2.5.31-debian_fixes-3.patch)*
- GCC Linkonce Patch - 12 Ko :  
*<http://www.linuxfromscratch.org/patches/lfs/6.1.1/gcc-3.4.3-linkonce-1.patch>*
- GCC No-Fixincludes Patch - 1 Ko :  
*[http://www.linuxfromscratch.org/patches/lfs/6.1.1/gcc-3.4.3-no\\_fixincludes-1.patch](http://www.linuxfromscratch.org/patches/lfs/6.1.1/gcc-3.4.3-no_fixincludes-1.patch)*
- GCC Specs Patch - 14 Ko :  
*<http://www.linuxfromscratch.org/patches/lfs/6.1.1/gcc-3.4.3-specs-2.patch>*
- Glibc Rtdl Search Dirs Patch - 1 Ko :  
*[http://www.linuxfromscratch.org/patches/lfs/6.1.1/glibc-2.3.4-rtdl\\_search\\_dirs-1.patch](http://www.linuxfromscratch.org/patches/lfs/6.1.1/glibc-2.3.4-rtdl_search_dirs-1.patch)*
- Glibc Fix Testsuite Patch - 1 Ko :  
*[http://www.linuxfromscratch.org/patches/lfs/6.1.1/glibc-2.3.4-fix\\_test-1.patch](http://www.linuxfromscratch.org/patches/lfs/6.1.1/glibc-2.3.4-fix_test-1.patch)*
- Glibc TLS Assertion Patch - 6 Ko :  
*[http://www.linuxfromscratch.org/patches/lfs/6.1.1/glibc-2.3.4-tls\\_assert-1.patch](http://www.linuxfromscratch.org/patches/lfs/6.1.1/glibc-2.3.4-tls_assert-1.patch)*
- Gzip Security Patch - 2 Ko :  
*[http://www.linuxfromscratch.org/patches/lfs/6.1.1/gzip-1.3.5-security\\_fixes-1.patch](http://www.linuxfromscratch.org/patches/lfs/6.1.1/gzip-1.3.5-security_fixes-1.patch)*
- Inetutils Kernel Headers Patch - 1 Ko :  
*[http://www.linuxfromscratch.org/patches/lfs/6.1.1/inetutils-1.4.2-kernel\\_headers-1.patch](http://www.linuxfromscratch.org/patches/lfs/6.1.1/inetutils-1.4.2-kernel_headers-1.patch)*

- Inetutils No-Server-Man-Pages Patch - 4 Ko :  
[http://www.linuxfromscratch.org/patches/lfs/6.1.1/inetutils-1.4.2-no\\_server\\_man\\_pages-1.patch](http://www.linuxfromscratch.org/patches/lfs/6.1.1/inetutils-1.4.2-no_server_man_pages-1.patch)
- Mktmp Tempfile Patch - 3 Ko :  
[http://www.linuxfromscratch.org/patches/lfs/6.1.1/mktemp-1.5-add\\_tempfile-2.patch](http://www.linuxfromscratch.org/patches/lfs/6.1.1/mktemp-1.5-add_tempfile-2.patch)
- Perl Libc Patch - 1 Ko :  
<http://www.linuxfromscratch.org/patches/lfs/6.1.1/perl-5.8.7-libc-1.patch>
- Readline Fixes Patch - 7 Ko :  
<http://www.linuxfromscratch.org/patches/lfs/6.1.1/readline-5.0-fixes-1.patch>
- Sysklogd Fixes Patch - 27 Ko :  
<http://www.linuxfromscratch.org/patches/lfs/6.1.1/sysklogd-1.4.1-fixes-1.patch>
- Tar Sparse Fix Patch - 1 Ko :  
[http://www.linuxfromscratch.org/patches/lfs/6.1.1/tar-1.15.1-sparse\\_fix-1.patch](http://www.linuxfromscratch.org/patches/lfs/6.1.1/tar-1.15.1-sparse_fix-1.patch)
- Texinfo Tempfile Fix Patch - 2 Ko :  
[http://www.linuxfromscratch.org/patches/lfs/6.1.1/texinfo-4.8-tempfile\\_fix-1.patch](http://www.linuxfromscratch.org/patches/lfs/6.1.1/texinfo-4.8-tempfile_fix-1.patch)
- Util-linux Cramfs Patch - 3 Ko :  
<http://www.linuxfromscratch.org/patches/lfs/6.1.1/util-linux-2.12q-cramfs-1.patch>
- Util-linux Umount Patch - 1 Ko :  
[http://www.linuxfromscratch.org/patches/lfs/6.1.1/util-linux-2.12q-umount\\_fix-1.patch](http://www.linuxfromscratch.org/patches/lfs/6.1.1/util-linux-2.12q-umount_fix-1.patch)
- Vim Security Patch - 8 Ko :  
[http://www.linuxfromscratch.org/patches/lfs/6.1.1/vim-6.3-security\\_fix-2.patch](http://www.linuxfromscratch.org/patches/lfs/6.1.1/vim-6.3-security_fix-2.patch)

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

## Chapitre 4. Dernières préparations

### 4.1. À propos de \$LFS

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

```
echo $LFS
```

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

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

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

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

## 4.2. Créer le répertoire `$LFS/tools`

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

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

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

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

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



### Note

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

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

## 4.3. Ajouter l'utilisateur LFS

Lorsque vous êtes connecté en tant qu'utilisateur *root*, faire une simple erreur peut endommager voire dévaster votre système. Donc, nous recommandons de construire les paquets dans ce chapitre en tant qu'utilisateur non privilégié. Vous pouvez bien sûr utiliser votre propre nom d'utilisateur mais, pour faciliter l'établissement d'un environnement de travail propre, créez un nouvel utilisateur nommé *lfs* comme membre d'un nouveau groupe *lfs* et utilisez-le lors du processus d'installation. En tant que *root*, lancez les commandes suivantes pour créer le nouvel utilisateur :

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

Voici la signification des options en ligne de commande :

*-s /bin/bash*

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

*-g lfs*

Cette option ajoute l'utilisateur *lfs* au groupe *lfs*.

*-m*

Ceci crée un répertoire personnel pour l'utilisateur *lfs*.

*-k /dev/null*

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

*lfs*

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

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

```
passwd lfs
```

Donnez-lui un accès complet à *\$LFS/tools* en indiquant que *lfs* est le propriétaire du répertoire :

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

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

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

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

```
su - lfs
```

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

## 4.4. Configurer l'environnement

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

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

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

La nouvelle instance du shell est un shell *sans connexion*, qui ne lit donc pas les fichiers `/etc/profile` et `.bash_profile` mais plutôt le fichier `.bashrc`. Créez le fichier `.bashrc` maintenant :

```
cat > ~/.bashrc << "EOF"
set +h
umask 022
LFS=/mnt/lfs
LC_ALL=POSIX
PATH=/tools/bin:/bin:/usr/bin
export LFS 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 `$LFS/tools` dès qu'ils sont disponibles et sans se rappeler de la version précédente du même programme mais dans un autre emplacement.

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

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

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

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

programmes sont disponibles depuis l'environnement du chapitre 5.



Enfin, pour avoir un environnement complètement préparé pour la construction des outils temporaires, récupérez le source du profile de l'utilisateur tout juste créé :

```
source ~/.bash_profile
```

## 4.5. À propos des SBU

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

La mesure SBU fonctionne ainsi. Le premier paquet lié statiquement, que vous compilez dans ce livre, est Binutils lors du Chapitre 5. Le temps que prend la compilation de ce paquet est ce que nous appelons « SBU ». Tous les autres temps de compilation sont exprimés relativement à ce temps.

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

En général, les SBU ne sont pas vraiment précis car ils dépendent de trop de facteurs, dont la version de GCC sur votre machine hôte. Notez que les SBU sont encore moins précis sur les machines multi-processeurs (SMP). Ils sont fournis ici pour donner une estimation du temps nécessaire pour installer un paquetage mais ces nombres peuvent varier de plusieurs dizaines de minutes dans certains cas.

Si vous souhaitez voir des chronométrages réels pour des machines spécifiques, nous recommandons la page d'accueil de « The LinuxFromScratch SBU » sur <http://www.linuxfromscratch.org/~bdubbs/>.

## 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 comme quoi tout a été compilé correctement. Une suite de tests réussissant l'ensemble des vérifications prouve généralement que le paquet fonctionne à peu près comme le développeur en avait l'intention. Néanmoins, cela ne garantit pas que le paquet ne contient pas de bogues.

Certaines des suites de tests sont plus importantes que d'autres. Par exemple, les suites de tests des paquets formant le cœur de l'ensemble des outils (GCC, Binutils et Glibc, la bibliothèque C) 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, spécialement sur du matériel lent, mais ces tests sont nécessaires.



### Note

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

Un problème commun lors du lancement des suites de test pour Binutils et GCC est de manquer de pseudo-terminaux (PTY). Le symptôme est un nombre inhabituellement haut de tests ayant échoués. Ceci peut arriver pour un certain nombre de raisons. La plus raisonnable est que le système hôte ne dispose pas d'un système de fichiers `devpts` file configuré correctement. Ce problème est discuté avec plus de détails dans le Chapitre 5.

Quelquefois, les suites de tests des paquets échoueront mais pour des raisons dont les développeurs sont conscients et qu'ils ont estimées non critiques. Consultez les traces situées dans <http://www.linuxfromscratch.org/lfs/build-logs/6.1.1/> pour vérifier si ces échecs sont attendus. Ce site est valide pour tous les tests effectués dans ce livre.

# Chapitre 5. Construire un système temporaire

## 5.1. Introduction

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

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

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



### Important

Avant de lancer les instructions de construction pour un paquet, le paquet doit être déballé en tant qu'utilisateur `lfs` et la commande `cd` doit être utilisé pour entrer dans le répertoire tout juste créé. Les instructions de construction supposent que le shell `bash` est utilisé.

Plusieurs paquets sont corrigés avant d'être compilés, mais seulement dans le cas où la correction est nécessaire pour résoudre un problème. Souvent, le correctif est nécessaire à la fois dans ce chapitre et dans le suivant, mais quelque fois dans seulement un des deux. Donc, ne vous inquiétez pas lorsque des instructions pour un correctif téléchargé semblent manquer. Des messages d'avertissements sur un décalage (*offset*) ou sur autre chose (*fuzz*) peuvent apparaître lors de l'application d'un correctif. Ne vous inquiétez pas pour ces messages, le correctif a bien été appliqué.

Pendant la compilation de la plupart des paquets, plusieurs messages d'avertissement du compilateur défileront sur votre écran. Ceci est normal et peut être ignoré sans danger. Ces messages d'avertissement ne sont que des avertissements—des avertissements sur un utilisation obsolète, mais pas invalide, de la syntaxe de C ou de C++. Les standards C changent assez souvent et quelques paquets continuent à utiliser les anciens standards. Ce n'est pas un véritable problème mais cela provoque les messages.



### Important

Après l'installation de chaque paquet, supprimez son répertoire source et son répertoire de construction, sauf si nous vous le demandons spécifiquement. Supprimer les sources empêche une mauvaise configuration lorsque le même paquet est réinstallé un peu plus tard. Seuls trois paquets ont besoin de conserver leur répertoire de sources et de construction pendant un moment pour que leur contenu soit utilisé par des commandes suivantes. Faites particulièrement attention à ces rappels.

Vérifiez une dernière fois que la variable d'environnement `LFS` est configurée correctement :

```
echo $LFS
```

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

## 5.2. Notes techniques sur l'ensemble d'outils

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

Le but global de Chapitre 5 est de fournir un environnement temporaire où nous pouvons utiliser chroot à partir duquel nous pouvons produire une construction propre, sans soucis, du système LFS cible dans Chapitre 6. Tout au long du chemin, nous nous séparons du système hôte autant que possible et, se faisant, nous construisons un ensemble d'outils qui se tient. Il devrait être noté que le processus de construction a été conçu pour minimiser les risques pour les nouveaux lecteurs et pour fournir une valeur éducative maximale en même temps.



### Important

Avant de continuer, faites attention au nom de la plateforme de travail, souvent appelé la triplète cible. De nombreuses fois, la triplète cible sera probablement *i686-pc-linux-gnu*. Une façon simple de déterminer le nom de la triplète cible est de lancer le script **config.guess** venant avec le source pour un grand nombre de paquetages. Déballez les sources de Binutils, lancez le script **./config.guess** et notez la sortie.

De même, faites attention au nom de l'éditeur de liens de la plateforme, souvent appelé le chargeur dynamique (à ne pas confondre avec l'éditeur de liens **ld** faisant partie de Binutils). Le chargeur dynamique fourni par Glibc trouve et charge les bibliothèques partagées nécessaires à un programme pour s'exécuter, puis l'exécute. Le nom de l'éditeur dynamique sera habituellement **ld-linux.so.2**. Sur des plateformes moins connues, le nom pourrait être **ld.so.1** alors que les nouvelles plateformes 64 bits pourraient être nommées encore différemment. Le nom de l'éditeur de liens dynamiques de la plateforme peut être déterminé en cherchant dans le répertoire **/lib** du système hôte. Une façon certaine de déterminer le nom est d'inspecter un binaire au hasard de la hôte système en exécutant : **readelf -l <nom du binaire> | grep interpreter** et de noter le résultat. La référence faisant autorité couvrant toutes les plateformes est dans le fichier **shlib-versions** à la racine du répertoire des sources de Glibc.

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

- Le processus est similaire dans son principe à la cross-compilation où les outils installés dans le même préfixe fonctionnent en coopération et utilisent, du coup, un peu de « magie » GNU
- Une manipulation attentionnée du chemin de recherche des bibliothèques de l'éditeur de liens standard vous assure que les programmes sont liés seulement avec les bibliothèques choisies
- Une manipulation attentionnée du fichier **specs** de **gcc** indique au compilateur l'éditeur de liens dynamique cible à utiliser

Binutils est tout d'abord installé parce que les exécutions de Glibc et GCC par **configure** réalisent quelques tests de fonctionnalités sur l'assembleur et l'éditeur de liens pour déterminer quelle fonctionnalité logicielle activer ou désactiver. Ceci est plus important que vous pourriez supposer. Un GCC ou une Glibc mal configurée peut résulter en un ensemble d'outils subtilement cassé, où l'impact d'une telle cassure ne se montrerait pas avant la fin de la construction de la distribution complète. Un échec dans la suite de tests surlignera habituellement sur cette erreur avant que trop de travail supplémentaire n'ait été réalisé.

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

Le prochain paquetage installé est GCC. Un exemple de ce qui peut être vu pendant son exécution de `configure` est :

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

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

Des informations détaillées peuvent être obtenues à partir de `gcc` en lui fournissant l'option en ligne de commande `-v` lors de la compilation d'un programme de tests. Par exemple, `gcc -v dummy.c` affichera des informations détaillées sur les étapes du préprocesseur, de la compilation et de l'assemblage, ceci incluant les chemins de recherche inclus par `gcc` et leur ordre.

Le prochain paquetage installé est Glibc. Les plus importantes considérations pour construire Glibc est le compilateur, les outils binaires et les en-têtes du noyau. Le compilateur n'est généralement pas un problème car Glibc utilise toujours le `gcc` trouvé dans un répertoire du `PATH`. Les outils binaires et les en-têtes du noyau peuvent être un peu plus compliqués. Du coup, ne prenez pas de risque et utilisez les options disponibles de `configure` pour renforcer les bonnes sélections. Après l'exécution de `configure`, vérifiez le contenu du fichier `config.make` dans le répertoire `glibc-build` pour tous les détails importants. Notez l'utilisation de `CC="gcc -B/tools/bin/"` pour contrôler les outils binaires utilisés, et l'utilisation de `-nostdinc` et `-isystem` pour contrôler le chemin de recherche des en-têtes du compilateur. Ces éléments soulignent un aspect important du paquetage Glibc — il est auto-suffisant en terme de machinerie de construction et ne se repose généralement pas sur l'ensemble d'outils par défaut.

Après l'installation de Glibc, réalisez les ajustements pour vous assurer que la recherche et l'édition de liens prennent seulement place à l'intérieur du préfixe `/tools`. Installez un `ld` ajusté qui a un chemin de recherche limité, codé en dur, vers `/tools/lib`. Puis, modifiez le fichier `specs` de `gcc` pour pointer vers le nouvel éditeur de liens dynamique dans `/tools/lib`. Cette dernière étape est vitale pour le processus complet. Comme mentionné ci-dessus, un chemin en dur vers un éditeur de liens est intégré dans chaque exécutable ainsi que dans chaque exécutable partagé (ELF). Ceci peut être inspecté en exécutant : `readelf -l <name of binary> | grep interpreter`. Modifier le fichier `specs` de `gcc` nous assure que chaque programme compilé à partir de maintenant et jusqu'à la fin de ce chapitre utilisera le nouvel éditeur de liens dynamiques dans `/tools/lib`.

Le besoin d'utiliser le nouvel éditeur de liens dynamique est aussi la raison pour laquelle le correctif `Specs` est appliqué lors de la seconde passe de GCC. Échouer sur ce point résultera en des programmes GCC ayant le nom de l'éditeur de liens provenant du répertoire `/lib` du système hôte intégré en eux, ce qui empêchera le but de s'éloigner de l'hôte.

Lors de la seconde passe de Binutils, nous sommes capable d'utiliser l'option `--with-lib-path` de `configure` pour contrôler le chemin de recherche des bibliothèques de `ld`. À partir de là, l'ensemble d'outils principal est contenu en lui-même. Le reste des paquetages de Chapitre 5 se construit à partir de la

nouvelle Glibc dans `/tools`.

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



## 5.3. Binutils-2.15.94.0.2.2 - Passe 1

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

**Temps de construction estimé :** 1,0 SBU

**Espace disque requis :** 179 Mo

**Dépendances de l'installation :** Bash, Bison, Coreutils, Diffutils, Flex, GCC, Gettext, Glibc, Grep, M4, Make, Perl, Sed et Texinfo

### 5.3.1. Installation de Binutils

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

Ce paquet est connu pour avoir des soucis quand les options d'optimisation par défaut (en incluant les options `-march` et `-mcpu`) sont modifiées. Donc, si des variables d'environnement qui surchargent les optimisations par défaut, telles que `CFLAGS` et `CXXFLAGS`, ont été définies, supprimez cette initialisation pour la construction de Binutils.

Si vous construisez à partir d'un hôte utilisant Gcc-4 ou ultérieur, il est nécessaire de corriger la première construction de cette version de Binutils pour qu'il puisse être compilé par le système hôte.

```
patch -Np1 -i ../binutils-2.15.94.0.2.2-gcc4-1.patch
```

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

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



#### Note

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

Maintenant, préparez la compilation de Binutils :

```
../binutils-2.15.94.0.2.2/configure --prefix=/tools --disable-nls
```

Voici la signification des options de configure :

`--prefix=/tools`

Ceci indique au script configure de se préparer à installer les programmes Binutils dans le répertoire `/tools`.

`--disable-nls`

Ceci désactive l'internationalisation car ce n'est pas nécessaire pour des outils temporaires.

Continuez avec la compilation du paquet :

```
make
```

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

Installez le paquet :

```
make install
```

Ensuite, préparez l'éditeur de liens pour la phase d'« ajustement » un peu plus tard :

```
make -C ld clean  
make -C ld LIB_PATH=/tools/lib
```

Voici la signification des paramètres de make :

```
-C ld clean
```

Ceci indique au programme make de supprimer tous les fichiers compilés du sous-répertoire `ld`.

```
-C ld LDFLAGS="-all-static" LIB_PATH=/tools/lib
```

Cette option reconstruit tout ce qui se trouve dans le sous-répertoire `ld`. Spécifier la variable `LIB_PATH` en ligne de commande du makefile nous autorise à écraser la valeur par défaut et à la faire pointer vers notre emplacement temporaire des outils. La valeur de cette variable spécifie le chemin de recherche des bibliothèques par défaut pour l'éditeur de liens. Cette préparation est utilisée plus tard dans le chapitre.



### **Avertissement**

*Ne supprimez pas encore* les répertoires de construction et des sources dont vous aurez encore besoin dans leur état actuel un peu plus tard dans ce chapitre.

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

## 5.4. GCC-3.4.3 - Passe 1

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

**Temps de construction estimé :** 4,4 SBU

**Espace disque requis :** 219 Mo

**Dépendances de l'installation :** Bash, Binutils, Coreutils, Diffutils, Findutils, Gawk, Gettext, Glibc, Grep, Make, Perl, Sed et Texinfo

### 5.4.1. Installation de GCC

Ce paquet est connu pour avoir des soucis quand les options d'optimisation par défaut (en incluant les options `-march` et `-mcpu`) sont modifiées. Donc, si des variables d'environnement qui surchargent les optimisations par défaut, telles que `CFLAGS` et `CXXFLAGS`, ont été définies, supprimez cette initialisation pour la construction de GCC.

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

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

Préparez la compilation de GCC :

```
../gcc-3.4.3/configure --prefix=/tools \
  --libexecdir=/tools/lib --with-local-prefix=/tools \
  --disable-nls --enable-shared --enable-languages=c
```

Voici la signification des options de configure :

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

Le but de cette option est de supprimer `/usr/local/include` du chemin de recherche des fichiers include de `gcc`. Ce n'est pas absolument essentiel ; néanmoins, c'est une aide pour minimiser l'influence du système hôte.

`--enable-shared`

Cette option permet la construction de `libgcc_s.so.1` et `libgcc_eh.a`. Disposer de `libgcc_eh.a` nous assure que le script configure de Glibc (le prochain paquet à compiler) produira de bons résultats.

`--enable-languages=c`

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

Continuez avec la compilation du paquet :

```
make bootstrap
```

Voici la signification des paramètres de make :

#### *bootstrap*

Cette cible ne compile pas GCC une seule fois mais plusieurs fois. Il utilise les programmes compilés dans le premier tour pour se compiler une deuxième fois, puis une troisième fois. Il compare alors les deuxième et troisième compilations pour s'assurer qu'il arrive à se reproduire lui-même sans fautes, ce qui semble vouloir dire qu'il a été compilé correctement.

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

Installez le paquet :

```
make install
```

En touche finale, créez un lien symbolique. Beaucoup de programmes et de scripts lancent **cc** au lieu de **gcc**, ceci pour conserver des programmes génériques et donc utilisables sur tout type de système Unix où le compilateur GNU C n'est pas toujours installé. Utiliser **cc** permet de libérer l'administrateur système dans son choix du compilateur C à installer.

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

Les détails sur ce paquet sont disponibles dans Section 6.14.2, « Contenu de GCC. »

## 5.5. Linux-Libc-Headers-2.6.11.2

Le paquet Linux-Libc-Headers contient les en-têtes du noyau « nettoyés ».

**Temps de construction estimé :** 0,1 SBU

**Espace disque requis :** 26,9 Mo

**Dépendances de l'installation :** Coreutils

### 5.5.1. Installation de Linux-Libc-Headers

Pendant des années, la pratique commune était d'utiliser les en-têtes « bruts » du noyau (directement de l'archive tar du noyau) dans `/usr/include` mais, au fil des ans, les développeurs du noyau ont pris fortement position contre cet état de fait. Ceci a donné naissance au projet Linux-Libc-Headers, qui a été conçu pour maintenir une version stable de l'interface de programmation des applications (API) des en-têtes du noyau Linux.

Installez les fichiers d'en-têtes :

```
cp -Rv include/asm-i386 /tools/include/asm
cp -Rv include/linux /tools/include
```

Si votre architecture n'est pas i386 (compatible), ajustez la première commande en accord.

Les détails sur ce paquet sont situés dans Section 6.9.2, « Contenu de Linux-Libc-Headers. »

## 5.6. Glibc-2.3.4

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

**Temps de construction estimé :** 11,8 SBU

**Espace disque requis :** 454 Mo

**Dépendances de l'installation :** Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Gettext, Grep, Make, Perl, Sed et Texinfo

### 5.6.1. Installation de Glibc

Ce paquet est connu pour avoir des soucis quand les options d'optimisation par défaut (en incluant les options *-march* et *-mcpu*) sont modifiées. Donc, si des variables d'environnement qui surchargent les optimisations par défaut, telles que *CFLAGS* et *CXXFLAGS*, ont été définies, supprimez cette initialisation pour la construction de Glibc.

Il devrait être noté que compiler Glibc de toute autre façon que celle proposée par le livre compromet la stabilité du système.

Glibc échoue sur deux tests lorsque le noyau en cours d'exécution est le 2.6.11 ou ultérieures. Le problème est dû aux tests eux-mêmes et n'a rien à voir avec la bibliothèque C ou le noyau. Si vous planifiez d'exécuter la suite de tests, appliquez ce correctif :

```
patch -Np1 -i ../glibc-2.3.4-fix_test-1.patch
```

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

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

Ensuite, préparez la compilation de Glibc :

```
../glibc-2.3.4/configure --prefix=/tools \
  --disable-profile --enable-add-ons \
  --enable-kernel=2.6.0 --with-binutils=/tools/bin \
  --without-gd --with-headers=/tools/include \
  --without-selinux
```

Voici la signification des options de configure :

*--disable-profile*

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

*--enable-add-ons*

Ceci indique à Glibc d'utiliser le composant NPTL comme bibliothèque de threads.

*--enable-kernel=2.6.0*

Ceci indique à Glibc de compiler la bibliothèque avec le support des noyaux 2.6.x.

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

Bien que pas nécessaire, ce commutateur nous assure qu'il ne reste aucune erreur provenant des programmes Binutils lors de la construction de Glibc.

`--without-gd`

Ce commutateur empêche la construction du programme **memusagestat**, programme qui insiste pour être lié avec les bibliothèques de l'hôte (libgd, libpng, libz et ainsi de suite).

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

Ceci indique à Glibc de se compiler lui-même avec les en-têtes récemment installés dans le répertoire des outils, de façon à ce qu'il sache exactement de quelles fonctionnalités disposent le noyau et qu'il puisse s'optimiser correctement.

`--without-selinux`

Lors de la construction à partir d'hôtes qui incluent la fonctionnalité SELinux (par exemple Fedora Core 3), Glibc construira le support pour SELinux. Comme l'environnement d'outils LFS ne contient pas de support pour SELinux, une Glibc compilée avec ce support ne fonctionnera pas correctement.

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

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

Le programme **msgfmt**, manquant ou incompatible, ne pose généralement pas de problème mais certaines personnes pensent qu'il peut poser problème lors de l'exécution de la suite de tests. Ce programme **msgfmt** fait partie du paquet Gettext que la distribution hôte devrait fournir. Si **msgfmt** est présent mais semble incompatible, mettez à jour le paquet Gettext du système hôte ou continuez sans et voyez si la suite de tests continue son exécution sans problèmes.

Compilez le paquet :

```
make
```

La compilation est maintenant terminée. Comme dit plus tôt, lancer les suites de tests pour les outils temporaires de ce chapitre n'est pas nécessaire. Pour exécuter la suite de tests Glibc, si désiré, lancer la commande suivante :

```
make check
```

Pour une discussion sur les échecs de tests qui ont une importance particulière, merci de voir Section 6.11, « Glibc-2.3.4. »

Dans ce chapitre, certains tests peuvent être perturbés par des outils existants ou par des problèmes d'environnement sur le système hôte. Les échecs de la suite de tests de Glibc dans ce chapitre ne portent typiquement pas à conséquence. La bibliothèque Glibc installée dans Chapitre 6 est celle qui sera utilisée à la fin, donc c'est celle qui aura besoin de passer la plupart des tests (y compris dans Chapitre 6, certains échecs pourraient survenir, par exemple celui des mathématiques).

Si vous expérimentez un échec, prenez-en note puis continuez en ré-exécutant la commande **make check**. La suite de tests devrait reprendre là où elle a quitté précédemment. Cette séquence d'arrêt/relancement est annulée en lançant la commande **make -k check**. En utilisant cette option, assurez-vous de tracer la sortie pour que le journal des traces puisse être examiné plus tard pour les différents échecs.

L'étape d'installation de Glibc affichera un message d'avertissement sans conséquence pour l'absence de `/tools/etc/ld.so.conf`. Supprimez ce message avec :

```
mkdir -v /tools/etc
touch /tools/etc/ld.so.conf
```

Installez le paquet :

```
make install
```

Différents pays et cultures ont des conventions variables sur la façon de communiquer. Ces conventions vont du très simple, telle que la représentation de la date et de l'heure à du très compliqué, telle que le langage parlé. L'internationalisation des programmes GNU fonctionne en utilisant les locales.



### Note

Si les suites de tests ne seront pas exécutés dans ce chapitre (suivant ainsi notre recommandation), il y a peu d'intérêts à installer les locales maintenant. Les bonnes locales seront installées dans le chapitre suivant.

Néanmoins, pour installer les locales Glibc, utilisez la commande suivante :

```
make localedata/install-locales
```

Pour gagner du temps, une alternative au lancement de la commande précédente (qui génère et installe chaque locale que Glibc connaît) est d'installer seulement les locales voulues et nécessaires. Ceci peut se faire en utilisant la commande **localedef**. Des informations sur cette commande sont situées dans le fichier `INSTALL` des sources de Glibc. Néanmoins, il existe un certain nombre de locales essentielles pour réussir les tests des futurs paquets, en particulier les tests de *libstdc++* pour GCC. Les instructions suivantes installeront l'ensemble minimale de locales nécessaires pour que les tests réussissent :

```
mkdir -pv /tools/lib/locale
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
```

Les détails sur ce paquet sont situés dans Section 6.11.4, « Contenu de Glibc. »



## 5.7. Ajuster l'atelier des outils

Maintenant que les bibliothèques C temporaires ont été installées, tous les outils compilés dans le reste de ce chapitre doivent être liés avec ces bibliothèques. Pour accomplir cela, l'éditeur de liens et le fichier specs du compilateur doivent être ajustés.

L'éditeur de liens ajusté (à la fin de la première passe de Binutils) est installé en lançant la commande suivante à partir du répertoire `binutils-build` :

```
make -C ld install
```

À partir de ce moment, tout sera lié uniquement avec les bibliothèques comprises dans `/tools/lib`.



### Note

Si l'avertissement précédent de différencier les répertoires source et de conversion de Binutils à partir de la première passe n'a pas été suivi, ignorez simplement la commande ci-dessus. Le résultat en est une petite chance de programmes de tests toujours liés avec les bibliothèques de l'hôte. Ce n'est pas idéal mais ce n'est pas un problème majeur. La situation est corrigée à l'installation de la deuxième passe de Binutils un peu plus tard.

Maintenant que l'éditeur de liens ajusté est installé, les répertoires source et de construction de Binutils doivent être supprimés.

La prochaine tâche est de modifier le fichier specs de GCC pour qu'il pointe vers le nouvel éditeur de liens. Un simple script sed devrait y parvenir :

```
SPECFILE=`gcc --print-file specs` &&
sed 's@ /lib/ld-linux.so.2@ /tools/lib/ld-linux.so.2@g' \
    $SPECFILE > tempspecfile &&
mv -f tempspecfile $SPECFILE &&
unset SPECFILE
```

Autrement, le fichier specs est éditable manuellement. Ceci est fait en remplaçant chaque occurrence de « `/lib/ld-linux.so.2` » par « `/tools/lib/ld-linux.so.2` ».

Assurez-vous d'inspecter visuellement le fichier specs pour vérifier que la modification attendue a été réellement réalisée.



### Important

Au cas où le nom de l'éditeur de liens de la plateforme de travail est autre que `ld-linux.so.2`, remplacez `ld-linux.so.2` avec le nom de l'éditeur de liens de votre plateforme dans les commandes ci-dessus. Référez-vous à Section 5.2, « Notes techniques sur l'ensemble d'outils, » si nécessaire.

Enfin, il existe un risque que certains fichiers include du système hôte aient trouvé leur chemin vers le répertoire include privé de GCC. Ceci peut arriver à cause du processus « `fixincludes` » de GCC fonctionnant en tant que partie de la construction GCC. Ceci est expliqué un peu plus tard dans ce chapitre. Lancez les commandes suivantes pour éliminer cette possibilité :

```
rm -vf /tools/lib/gcc/*/*/include/{pthread.h,bits/sigthread.h}
```



## Attention

Il est impératif à ce moment de s'arrêter et de s'assurer que les fonctions basiques (compilation et édition des liens) du nouvel atelier d'outils fonctionnent comme attendu. Pour réaliser une vérification de santé, lancez les commandes suivantes :

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

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

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

Notez que `/tools/lib` apparaît comme préfixe du chargeur dynamique.

Si l'affichage diffère ou s'il n'y a aucun affichage, alors quelque chose ne se passe pas bien. Investiguez et retracez vos étapes pour trouver où se cache le problème et comment le corriger. Ce problème doit être corrigé avant de continuer. Tout d'abord, relancez la vérification en utilisant `gcc` au lieu de `cc`. Si cela fonctionne, le lien symbolique `/tools/bin/cc` est manquant. Revisitez Section 5.4, « GCC-3.4.3 - Passe 1, » et installez le lien symbolique. Ensuite, assurez-vous que le `PATH` est correct. Ceci se vérifie en lançant `echo $PATH` et en vérifiant que `/tools/bin` est en tête de la liste. Si le `PATH` est mauvais, cela pourrait signifier que vous n'êtes pas connecté en tant qu'utilisateur `lfs` ou que quelque chose s'est mal passé dans Section 4.4, « Configurer l'environnement. » Une autre possibilité est que quelque chose a pu mal se passer avec la correction du fichier specs ci-dessus. Dans ce cas, refaites la modification de ce fichier.

Une fois satisfait, nettoyez les fichiers de test :

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

Construire TCL dans la prochaine section servira comme vérification supplémentaire de la bonne mise en place de l'outil de construction. Si TCL échoue à la construction, c'est une indication d'un problème avec l'installation de Binutils, GCC ou Glibc, mais pas avec TCL lui-même.

## 5.8. Tcl-8.4.9

Le paquet Tcl contient le langage de commandes des outils (*Tool Command Language*).

**Temps de construction estimé :** 0,9 SBU

**Espace disque requis :** 23,3 Mo

**Dépendances de l'installation :** Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Grep, Make et Sed

### 5.8.1. Installation de Tcl

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

Préparez la compilation de Tcl :

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

Construisez le paquet :

```
make
```

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

Installez le paquet :

```
make install
```



#### Avertissement

*Ne supprimez pas encore* le répertoire des sources `tcl8.4.9` car le paquet suivant a besoin des en-têtes.

Initialisez une variable avec le chemin complet du répertoire actuel. Le prochain paquetage, Expect, utilisera cette variable pour trouver les en-têtes de Tcl.

```
cd ..
export TCLPATH=`pwd`
```

Maintenant, créez un lien symbolique nécessaire :

```
ln -sv tclsh8.4 /tools/bin/tclsh
```

## 5.8.2. Contenu de Tcl

**Programmes installés:** tclsh (lien vers tclsh8.4) et tclsh8.4

**Bibliothèque installée:** libtcl8.4.so

### Descriptions courtes

**tclsh8.4**            Le shell de commandes Tcl

**tclsh**              Un lien vers tclsh8.4

**libtcl8.4.so**    La bibliothèque Tcl

## 5.9. Expect-5.43.0

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

**Temps de construction estimé :** 0,1 SBU

**Espace disque requis :** 4,0 Mo

**Dépendances de l'installation :** Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Grep, Make, Sed et Tcl

### 5.9.1. Installation d'Expect

Tout d'abord, corrigez un bogue résultant en de nombreux faux échecs lors de l'exécution de la suite de tests de GCC :

```
patch -Np1 -i ../expect-5.43.0-spawn-1.patch
```

Maintenant, préparez la compilation d'Expect :

```
./configure --prefix=/tools --with-tcl=/tools/lib \
  --with-tclinclude=$TCLPATH --with-x=no
```

Voici la signification des options de configure :

*--with-tcl=/tools/lib*

Ceci nous assure que le script configure trouve l'installation Tcl dans l'emplacement temporaire des outils à la place d'un résidant sur le système hôte.

*--with-tclinclude=\$TCLPATH*

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

*--with-x=no*

Ceci indique au script configure de ne pas chercher Tk (le composant interface de Tcl) ou les bibliothèques d'X Window System, les deux pouvant résider sur le système hôte mais n'existant pas sur l'environnement temporaire.

Construisez le paquet :

```
make
```

Pour tester les résultats, lancez : **make test**. Notez que la suite de tests d'Expect est connue pour avoir de nombreux échecs sous certaines conditions de l'hôte, conditions qui ne sont pas de notre contrôle. Du coup, les échecs de la suite de tests ne sont pas surprenantes et ne sont pas considérés comme critiques.)

Installez-le :

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

Voici la signification du paramètre de make :

*SCRIPTS=""*

Ceci empêche l'installation de scripts expect supplémentaires non nécessaires.

Maintenant, supprimez la variable TCLPATH :

```
unset TCLPATH
```

Les répertoires des sources de Tcl et d'Expect peuvent maintenant être supprimés.

## 5.9.2. Contenu d'Expect

**Programme installé:** expect

**Bibliothèque installée:** libexpect-5.43.a

### Descriptions courtes

<b>expect</b>	Communique avec les autres programmes interactifs suivant un script.
<code>libexpect-5.43.a</code>	Contient des fonctions qui permettent à Expect d'être utilisé comme une extension Tcl ou d'être utilisé directement à partir du langage C ou du langage C++ (sans Tcl)

## 5.10. DejaGNU-1.4.4

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

**Temps de construction estimé :** 0,1 SBU

**Espace disque requis :** 6,1 Mo

**Dépendances de l'installation :** Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Grep, Make et Sed

### 5.10.1. Installation de DejaGNU

Préparez la compilation de DejaGNU :

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

Construisez et installez le paquet :

```
make install
```

### 5.10.2. Contenu de DejaGNU

**Programme installé:** runtest

#### Descriptions courtes

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

## 5.11. GCC-3.4.3 - Passe 2

**Temps de construction estimé :** 11,0 SBU

**Espace disque requis :** 292 Mo

**Dépendances de l'installation :** Bash, Binutils, Coreutils, Diffutils, Findutils, Gawk, Gettext, Glibc, Grep, Make, Perl, Sed et Texinfo

### 5.11.1. Re-installation de GCC

Ce paquet est connu pour avoir des soucis quand les options d'optimisation par défaut (en incluant les options `-march` et `-mcpu`) sont modifiées. Donc, si des variables d'environnement qui surchargent les optimisations par défaut, telles que `CFLAGS` et `CXXFLAGS`, ont été définies, supprimez cette initialisation pour la construction de GCC.

Les outils requis pour tester GCC et Binutils (Tcl, Expect et DejaGnu) sont maintenant installés. GCC et Binutils peuvent maintenant être reconstruit en les liant avec la nouvelle Glibc et en les testant correctement (si vous souhaitez lancer les suites de tests dans ce chapitre). Merci de noter que ces suites de tests dépendent énormément de pseudos terminaux (PTY) fonctionnels fournis par votre distribution hôte. Les PTY sont le plus souvent implémentés via le système de fichiers `devpts`. Vérifiez si le système hôte est correctement configuré en réalisant un simple test :

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

Le résultat devrait être :

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

Si vous récupérez le message ci-dessus, la distribution hôte n'est pas correctement configurée pour les PTY. Dans ce cas, il ne sert à rien de lancer les suites de tests de GCC et Binutils jusqu'à la correction de ce problème. Merci de consulter la FAQ LFS sur <http://www.linuxfromscratch.org/lfs/faq.html#no-ptys> pour plus d'informations sur la façon de faire fonctionner les PTY.

Tout d'abord, corrigez un problème connu et faites un ajustement essentiel :

```
patch -Np1 -i ../gcc-3.4.3-no_fixincludes-1.patch
patch -Np1 -i ../gcc-3.4.3-specs-2.patch
```

Le premier correctif désactive le script GCC **fixincludes**. Ceci a déjà été mentionné brièvement mais une explication plus en détail de **fixincludes** est apportée ici. Sous des circonstances normales, le script GCC **fixincludes** parcourt le système pour trouver les fichiers d'en-tête qui ont besoin d'être corrigé. Il pourrait trouver que certains des fichiers d'en-têtes de Glibc sur le système devraient être corrigés, les corriger et les placer dans le répertoire des en-têtes privés de GCC. Dans le Chapitre 6, après avoir installé la nouvelle Glibc, ce répertoire serait recherché avant le répertoire `include` du système, faisant que GCC trouverait les en-têtes corrigés du système hôte qui ne correspondront certainement pas à la version de Glibc actuellement utilisée pour le système LFS.

Le deuxième correctif modifie l'emplacement par défaut de l'éditeur de liens dynamiques de GCC (généralement `ld-linux.so.2`). Il supprime aussi `/usr/include` du chemin de recherche des `includes` de GCC. Corriger maintenant plutôt qu'ajuster le fichier `specs` après l'installation nous assure que l'éditeur de liens dynamiques sera utilisé lors de la construction de GCC. C'est-à-dire, tous les binaires finaux (et temporaires) créés lors de la construction seront liés à la nouvelle Glibc.



**Important**

Les correctifs ci-dessus sont critiques pour s'assurer une construction avec succès. N'oubliez pas de les appliquer.

De nouveau, créez un répertoire de construction séparé :

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

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

Maintenant, préparez la compilation de GCC :

```
../gcc-3.4.3/configure --prefix=/tools \
  --libexecdir=/tools/lib --with-local-prefix=/tools \
  --enable-clocale=gnu --enable-shared \
  --enable-threads=posix --enable-__cxa_atexit \
  --enable-languages=c,c++ --disable-libstdcxx-pch
```

Voici la signification des nouvelles options de configure :

`--enable-clocale=gnu`

Cette option nous assure que le bon modèle de locale est sélectionné pour les bibliothèques C++ sous toutes les circonstances. Si le script configure trouve la locale *de\_DE* installée, il sélectionnera le bon modèle de locale gnu. Néanmoins, si la locale *de\_DE* n'est pas installée, il existe un risque de construire des bibliothèques C++ incompatibles avec ABI à cause du choix d'un mauvais modèle générique de locale.

`--enable-threads=posix`

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

`--enable-__cxa_atexit`

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

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

Cette option est nécessaire pour s'assurer que les compilateurs C et C++ seront construits.

`--disable-libstdcxx-pch`

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

Compilez le paquet :

```
make
```

Il n'est pas nécessaire d'utiliser la cible *bootstrap* maintenant car le compilateur utilisé pour compiler ce GCC a été construit avec exactement la même version des sources de GCC utilisées précédemment.

La compilation est maintenant terminée. Comme mentionné plus tôt, lancer les suites de test pour les outils temporaires de ce chapitre n'est pas nécessaire. Néanmoins, pour exécuter la suite de tests de GCC, lancez la commande suivante :

```
make -k check
```

L'option `-k` est utilisée pour faire en sorte que toute la suite de tests est exécutée et qu'elle ne s'arrête pas au premier échec. La suite de tests GCC est très complète et il est pratiquement garanti que certaines erreurs apparaîtront. Pour obtenir un résumé des résultats de la suite de tests, lancez ceci :

```
../gcc-3.4.3/contrib/test_summary
```

Pour un simple résumé, envoyez la sortie sur un tube suivi de **grep -A7 Summ.**

Les résultats peuvent être comparés à ceux postés sur <http://www.linuxfromscratch.org/lfs/build-logs/6.1.1/>.

Quelques échecs inattendus ne peuvent souvent pas être évités. Les développeurs GCC sont généralement au courant mais ne les ont pas encore résolus. À moins que vos tests soient grandement différents de ceux de l'URL ci-dessus, vous pouvez continuer sans crainte.

Installez le paquet :

```
make install
```



### Note

À ce moment, il est fortement recommandé de répéter la vérification que nous avons réalisé dans ce chapitre. Référez-vous à Section 5.7, « Ajuster l'atelier des outils » et répétez le test de compilation. Si les résultats sont mauvais, alors la raison probable en est l'oubli de l'application du correctif "GCC Specs" mentionné ci-dessus.

Les détails sur ce paquet sont situés dans Section 6.14.2, « Contenu de GCC. »

## 5.12. Binutils-2.15.94.0.2.2 - Passe 2

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

**Temps de construction estimé :** 1,5 SBU

**Espace disque requis :** 114 Mo

**Dépendances de l'installation :** Bash, Bison, Coreutils, Diffutils, Flex, GCC, Gettext, Glibc, Grep, M4, Make, Perl, Sed et Texinfo

### 5.12.1. Ré-installation de Binutils

Ce paquet est connu pour avoir des soucis quand les options d'optimisation par défaut (en incluant les options `-march` et `-mcpu`) sont modifiées. Donc, si des variables d'environnement qui surchargent les optimisations par défaut, telles que `CFLAGS` et `CXXFLAGS`, ont été définies, supprimez cette initialisation pour la construction de Binutils.

Créez de nouveau un répertoire de construction séparé :

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

Préparez la compilation de Binutils :

```
../binutils-2.15.94.0.2.2/configure --prefix=/tools \
  --disable-nls --enable-shared --with-lib-path=/tools/lib
```

Voici la signification des nouvelles options de configure :

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

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

Compilez le paquet :

```
make
```

La compilation est maintenant terminée. Comme dit plutôt, lancer les suites de tests n'est pas nécessaire pour les outils temporaires dans ce chapitre. Néanmoins, pour lancer la suite de tests Binutils, lancez la commande suivante :

```
make check
```

Installez le paquet :

```
make install
```

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

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



**Avertissement**

*Ne supprimez pas encore* les répertoires des sources et de construction de Binutils. Ces répertoires seront nécessaires pour le prochain chapitre dans l'état où ils sont actuellement.

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

## 5.13. Gawk-3.1.4

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

**Temps de construction estimé :** 0,2 SBU

**Espace disque requis :** 16,4 Mo

**Dépendances de l'installation :** Bash, Binutils, Coreutils, Diffutils, GCC, Gettext, Glibc, Grep, Make et Sed

### 5.13.1. Installation de Gawk

Préparez la compilation de Gawk :

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

Compilez le paquet :

```
make
```

Pour tester les résultats, lancez : **make check**.

Installez le paquet :

```
make install
```

Les détails sur ce paquet sont situés dans Section 6.20.2, « Contenu de Gawk. »

## 5.14. Coreutils-5.2.1

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

**Temps de construction estimé :** 0,9 SBU

**Espace disque requis :** 53,3 Mo

**Dépendances de l'installation :** Bash, Binutils, Coreutils, Diffutils, GCC, Gettext, Glibc, Grep, Make, Perl et Sed

### 5.14.1. Installation de Coreutils

Préparez la compilation de Coreutils :

```
DEFAULT_POSIX2_VERSION=199209 ./configure --prefix=/tools
```

Ce paquet a un problème lorsqu'il est compilé avec des versions de Glibc plus anciennes que la 2.3.2. Certains outils de Coreutils (tels que **head**, **tail** et **sort**) rejettent leur syntaxe traditionnelle, une syntaxe utilisée depuis environ 30 ans. Cette ancienne syntaxe est si ancrée que la compatibilité doit être préservée jusqu'à ce que les endroits où elle est utilisée pourront être mis à jour. La compatibilité descendante est obtenue en initialisant la variable d'environnement `DEFAULT_POSIX2_VERSION` à « 199209 » dans la commande ci-dessus. Si vous ne voulez pas que Coreutils soit compatible avec la syntaxe traditionnelle, oubliez simplement d'initialiser la variable d'environnement `DEFAULT_POSIX2_VERSION`. Il est important de se rappeler que faire ceci aura des conséquences, dont la correction des nombreux paquets utilisant toujours l'ancienne syntaxe. Il est donc fortement recommandé de suivre exactement les instructions comme indiquées ci-dessus.

Compilez le paquet :

```
make
```

Pour tester les résultats, lancez : **make RUN\_EXPENSIVE\_TESTS=yes check**. Le paramètre `RUN_EXPENSIVE_TESTS=yes` indique à la suite de tests de lancer quelques tests supplémentaires, considérés relativement coûteux (en terme de puissance CPU et d'utilisation mémoire) mais habituellement sans problème sous Linux.

Installez le paquet :

```
make install
```

Les détails sur ce paquet sont disponibles dans Section 6.15.2, « Contenu de Coreutils. ».

## 5.15. Bzip2-1.0.3

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

**Temps de construction estimé** : 0,1 SBU

**Espace disque requis** : 3,5 Mo

**Dépendances de l'installation** : Bash, Binutils, Coreutils, Diffutils, GCC, Glibc et Make

### 5.15.1. Installation de Bzip2

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

```
make
```

Installez le paquet :

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

Les détails sur ce paquet sont situés dans Section 6.40.2, « Contenu de Bzip2. »

## 5.16. Gzip-1.3.5

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

**Temps de construction estimé :** 0,1 SBU

**Espace disque requis :** 2,2 Mo

**Dépendances de l'installation :** Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Grep, Make et Sed

### 5.16.1. Installation de Gzip

Préparez la compilation de Gzip :

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

Compilez le paquet :

```
make
```

Ce paquet ne dispose pas de suite de tests.

Installez le paquet :

```
make install
```

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



## 5.17. Diffutils-2.8.1

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

**Temps de construction estimé :** 0,1 SBU

**Espace disque requis :** 5,6 Mo

**Dépendances de l'installation :** Bash, Binutils, Coreutils, Diffutils, GCC, Gettext, Glibc, Grep, Make et Sed

### 5.17.1. Installation de Diffutils

Préparez la compilation de Diffutils :

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

Compilez le paquet :

```
make
```

Ce paquet ne contient pas de suite de tests.

Installez le paquet :

```
make install
```

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

## 5.18. Findutils-4.2.23

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

**Temps de construction estimé :** 0,2 SBU

**Espace disque requis :** 8,9 Mo

**Dépendances de l'installation :** Bash, Binutils, Coreutils, Diffutils, GCC, Gettext, Glibc, Grep, Make et Sed

### 5.18.1. Installation de Findutils

Préparez la compilation de Findutils :

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

Compilez le paquet :

```
make
```

Pour tester les résultats, lancez : **make check**.

Installez le paquet :

```
make install
```

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

## 5.19. Make-3.80

Le paquet Make contient un programme pour compiler des paquetages.

**Temps de construction estimé :** 0,2 SBU

**Espace disque requis :** 7,1 Mo

**Dépendances de l'installation :** Bash, Binutils, Coreutils, Diffutils, GCC, Gettext, Glibc, Grep et Sed

### 5.19.1. Installation de Make

Préparez la compilation de Make :

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

Compilez le paquet :

```
make
```

Pour tester les résultats, lancez : **make check**.

Installez le paquet :

```
make install
```

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

## 5.20. Grep-2.5.1a

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

**Temps de construction estimé** : 0,1 SBU

**Espace disque requis** : 4,5 Mo

**Dépendances de l'installation** : Bash, Binutils, Coreutils, Diffutils, GCC, Gettext, Glibc, Make, Sed et Texinfo

### 5.20.1. Installation de Grep

Préparez la compilation de Grep :

```
./configure --prefix=/tools \
--disable-perl-regexp
```

Voici la signification des options de configure :

*--disable-perl-regexp*

Ceci nous assure que **grep** ne sera pas lié à une bibliothèque PCRE qui pourrait être présente sur l'hôte et qui ne serait pas disponible une fois que nous serons entrés dans l'environnement **chroot**.

Compilez les programmes :

```
make
```

Pour tester les résultats, lancez : **make check**.

Installez le paquet :

```
make install
```

Les détails sur ce paquet sont disponibles dans Section 6.44.2, « Contenu de Grep ».

## 5.21. Sed-4.1.4

Le paquet Sed contient un éditeur de flux.

**Temps de construction estimé** : 0,2 SBU

**Espace disque requis** : 8,4 Mo

**Dépendances de l'installation** : Bash, Binutils, Coreutils, Diffutils, GCC, Gettext, Glibc, Grep, Make et Texinfo

### 5.21.1. Installation de Sed

Préparez la compilation de Sed :

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

Compilez le paquet :

```
make
```

Pour tester les résultats, lancez : **make check**.

Installez le paquet :

```
make install
```

Les détails de ce paquet sont situés dans Section 6.28.2, « Contenu de Sed. »

## 5.22. Gettext-0.14.3

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

**Temps de construction estimé :** 0,5 SBU

**Espace disque requis :** 63 Mo

**Dépendances de l'installation :** Bash, Binutils, Bison, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Make et Sed

### 5.22.1. Installation de Gettext

Préparez la compilation de Gettext :

```
./configure --prefix=/tools --disable-libasprintf \
--without-csharp
```

Voici la signification des options de configure :

*--disable-libasprintf*

Ce commutateur indique à Gettext de ne pas construire la bibliothèque `asprintf`. Parce que rien dans ce chapitre ou le suivant ne requiert cette bibliothèque et que Gettext est reconstruit plus tard, l'exclure sauve du temps et de l'espace.

*--without-csharp*

Ceci nous assure que Gettext ne construira pas le support du compilateur C# qui pourrait être présent sur l'hôte mais qui ne sera pas disponible une fois que nous serons entrés dans l'environnement du **chroot**.

Compilez le paquet :

```
make
```

Pour tester les résultats, lancez : **make check**. Ceci peut prendre beaucoup de temps, environ 7 SBU. La suite de tests Gettext est connue pour avoir des échecs sous certaines conditions liées à l'hôte, par exemple lorsqu'il trouve un compilateur Java sur l'hôte. Un correctif expérimental désactivant Java est disponible à partir du projet des correctifs LFS sur <http://www.linuxfromscratch.org/patches/>.

Installez le paquet :

```
make install
```

Les détails sur ce paquet sont situés dans Section 6.30.2, « Contenu de Gettext. »

## 5.23. Ncurses-5.4

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

**Temps de construction estimé :** 0,7 SBU

**Espace disque requis :** 27,5 Mo

**Dépendances de l'installation :** Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Make et Sed

### 5.23.1. Installation de Ncurses

Préparez la compilation de Ncurses :

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

Voici la signification des options de configure :

*--without-ada*

Ceci nous assure que Ncurses ne construira pas le support du compilateur Ada qui pourrait être présent sur l'hôte mais qui ne sera pas disponible lorsque nous entrerons dans l'environnement **chroot**.

*--enable-overwrite*

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

Compilez le paquet :

```
make
```

Ce paquet ne fournit pas de suite de tests.

Installez le paquet :

```
make install
```

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

## 5.24. Patch-2.5.4

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

**Temps de construction estimé** : 0,1 SBU

**Espace disque requis** : 1,5 Mo

**Dépendances de l'installation** : Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Grep, Make et Sed

### 5.24.1. Installation de Patch

Préparez la compilation de Patch :

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

Le commutateur du préprocesseur `-D_GNU_SOURCE` n'est nécessaire que sur la plateforme PowerPC. Il peut être oublié pour les autres.

Compilez le paquet :

```
make
```

Ce paquet ne fournit pas de suite de tests.

Installez le paquet :

```
make install
```

Les détails sur ce paquet sont situés dans Section 6.51.2, « Contenu de Patch. »



## 5.25. Tar-1.15.1

Le paquet Tar contient un programme d'archivage.

**Temps de construction estimé :** 0,2 SBU

**Espace disque requis :** 12,7 Mo

**Dépendances de l'installation :** Bash, Binutils, Coreutils, Diffutils, GCC, Gettext, Glibc, Grep, Make et Sed

### 5.25.1. Installation de Tar

Préparez la compilation de Tar :

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

Compilez le paquet :

```
make
```

Pour tester les résultats, lancez : **make check**.

Installez le paquet :

```
make install
```

Les détails sur ce paquet sont situés dans Section 6.57.2, « Contenu de Tar. »

## 5.26. Texinfo-4.8

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

**Temps de construction estimé :** 0,2 SBU

**Espace disque requis :** 14,7 Mo

**Dépendances de l'installation :** Bash, Binutils, Coreutils, Diffutils, GCC, Gettext, Glibc, Grep, Make, Ncurses et Sed

### 5.26.1. Installation de Texinfo

Préparez la compilation de Texinfo :

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

Compilez le paquet :

```
make
```

Pour tester les résultats, lancez : **make check**.

Installez le paquet :

```
make install
```

Les détails de ce paquet sont situés dans Section 6.34.2, « Contenu de Texinfo. »

## 5.27. Bash-3.0

Le paquet Bash contient le shell Bourne-Again.

**Temps de construction estimé :** 1,2 SBU

**Espace disque requis :** 20,7 Mo

**Dépendances de l'installation :** Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Make, Ncurses et Sed.

### 5.27.1. Installation de Bash

Bash a un problème lorsqu'il est compilé avec les nouvelles versions de Glibc, le faisant s'arrêter brutalement. Cette commande corrige ce problème :

```
patch -Np1 -i ../bash-3.0-avoid_WCONTINUED-1.patch
```

Préparez la compilation de Bash :

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

Voici la signification des options de configure :

*--without-bash-malloc*

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

Compilez le paquet :

```
make
```

Pour tester les résultats, lancez : **make tests**.

Installez le paquet :

```
make install
```

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

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

Les détails sur ce paquet sont situés dans Section 6.37.2, « Contenu de Bash. »

## 5.28. M4-1.4.3

Le paquet M4 contient un processeur de macros.

**Temps de construction estimé :** 0,1 SBU

**Espace disque requis :** 2,8 Mo

**Dépendances de l'installation :** Bash, Binutils, Coreutils, Diffutils, GCC, Gettext, Glibc, Grep, Make, Perl et Sed

### 5.28.1. Installation de M4

Préparez la compilation de M4 :

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

Compilez le paquetage :

```
make
```

Pour tester les résultats, lancez : **make check**.

Installez le paquetage :

```
make install
```

Les détails sur ce paquetage sont situés dans Section 6.24.2, « Contenu de M4. »

## 5.29. Bison-2.0

Le paquet Bison contient un générateur d'analyseurs.

**Temps de construction estimé :** 0,6 SBU

**Espace disque requis :** 10,0 Mo

**Dépendances de l'installation :** Bash, Binutils, Coreutils, Diffutils, GCC, Gettext, Glibc, Grep, M4, Make et Sed

### 5.29.1. Installation de Bison

Préparez la compilation de Bison :

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

Compilez le paquet :

```
make
```

Pour tester les résultats, lancez : **make check**.

Installez le paquet :

```
make install
```

Les détails sur ce paquet sont situés dans Section 6.25.2, « Contenu de Bison. »

## 5.30. Flex-2.5.31

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

**Temps de construction estimé :** 0,6 SBU

**Espace disque requis :** 22,5 Mo

**Dépendances de l'installation :** Bash, Binutils, Bison, Coreutils, Diffutils, GCC, Gettext, Glibc, Grep, M4, Make et Sed

### 5.30.1. Installation de Flex

Flex contient quelques bogues connus. Ils sont corrigés avec le correctif suivant :

```
patch -Np1 -i ../flex-2.5.31-debian_fixes-3.patch
```

GNU autotools détectera que le code source de Flex a été modifié par le correctif précédent et tente de mettre à jour la page man en accord. Ceci ne fonctionne pas sur certains systèmes et la page par défaut est suffisante, donc assurez-vous qu'elle n'est pas régénérée :

```
touch doc/flex.1
```

Maintenant, préparez la compilation de Flex :

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

Compilez le paquet :

```
make
```

Pour tester les résultats, lancez : **make check**.

Installez le paquet :

```
make install
```

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

## 5.31. Util-linux-2.12q

Le paquet Util-linux contient différents outils. Parmi eux se trouvent des outils de gestion des systèmes de fichiers, de consoles, de partitions et des messages.

**Temps de construction estimé :** 0,2 SBU

**Espace disque requis :** 8,9 Mo

**Dépendances de l'installation :** Bash, Binutils, Coreutils, Diffutils, GCC, Gettext, Glibc, Grep, Make, Ncurses, Sed et Zlib

### 5.31.1. Installation de Util-linux

Util-linux n'utilise pas les en-têtes et bibliothèques tout juste installés dans le répertoire `/tools` par défaut. Ceci est corrigé en modifiant le script configure :

```
sed -i 's@/usr/include@/tools/include@g' configure
```

Préparez la compilation de Util-linux :

```
./configure
```

Compilez quelques routines d'aide :

```
make -C lib
```

Seuls quelques utilitaires contenus dans ce paquet sont nécessaires :

```
make -C mount mount umount
make -C text-utils more
```

Ce paquet ne fournit pas de suite de tests.

Copiez ces programmes dans le répertoire des outils temporaires: :

```
cp mount/{,u}mount text-utils/more /tools/bin
```

Les détails sur ce paquet sont disponibles dans Section 6.59.3, « Contenu d'Util-linux. »

## 5.32. Perl-5.8.7

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

**Temps de construction estimé :** 0,8 SBU

**Espace disque requis :** 81,6 Mo

**Dépendances de l'installation :** Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Make et Sed

### 5.32.1. Installation de Perl

Tout d'abord, adaptez quelques chemins codés en dur vers la bibliothèque C en appliquant le correctif suivant :

```
patch -Np1 -i ../perl-5.8.7-libc-1.patch
```

Préparez la compilation de Perl (assurez-vous que la partie de la commande marquée « IO Fcntl POSIX » est saisie correctement, ce ne sont que des lettres) :

```
./configure.gnu --prefix=/tools -Dstatic_ext='IO Fcntl POSIX'
```

Voici la signification de l'option de configure :

```
-Dstatic_ext='IO Fcntl POSIX'
```

Ceci indique à Perl de construire l'ensemble minimal d'extensions statiques nécessaires à l'installation et au test du paquet Coreutils dans le prochain chapitre.

Seulement une partie des outils de ce paquetage doit être construit :

```
make perl utilities
```

Bien que Perl est fourni avec une suite de tests, il n'est pas recommandé de l'exécuter maintenant. Seules des parties de Perl ont été construites et l'exécution de **make test** obligerait la construction du reste de Perl, ce qui n'est pas nécessaire actuellement. La suite de tests peut être exécuté dans le chapitre suivant si désiré.

Puis, installez ces outils et leurs bibliothèques :

```
cp -v perl pod/pod2man /tools/bin
mkdir -pv /tools/lib/perl5/5.8.7
cp -Rv lib/* /tools/lib/perl5/5.8.7
```

Les détails sur ce paquet sont disponibles dans Section 6.33.2, « Contenu de Perl. »



## 5.33. Supprimer les symboles des fichiers objets

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

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

La dernière des commandes ci-dessus laissera de côté une vingtaine de fichiers en indiquant qu'elle ne reconnaît pas leur format. La plupart sont des scripts et non pas des binaires.

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

Pour sauver encore 30 Mo, supprimez toute la documentation :

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

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

# Construction du système LFS

# Chapitre 6. Installer les logiciels du système de base

## 6.1. Introduction

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

L'installation du logiciel est simple. Bien que, dans beaucoup de cas, les instructions d'installation pourraient être plus courtes et plus génériques, nous avons opté pour fournir les instructions complètes pour chaque paquet et minimiser ainsi les possibilités d'erreurs. La clé pour apprendre ce qui fait fonctionner un système Linux est de savoir à quoi sert chaque paquet et pourquoi l'utilisateur (ou le système) en a besoin. Pour chaque paquet installé, un résumé de son contenu est donné, suivant par des descriptions concises de chaque programme et de chaque bibliothèque que le paquet a installé.

En utilisant les optimisations du compilateur fournies dans ce chapitre, merci de lire l'astuce sur l'optimisation sur <http://www.linuxfromscratch.org/hints/downloads/files/optimization.txt>. Les optimisations du compilateur peuvent faire qu'un programme s'exécute 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 pourrait avoir été mal compilé à cause des interactions complexes entre le code et les outils de construction. Le petit potentiel de gains réussi en utilisant les optimisations de compilation est souvent ridicule comparé aux risques. Les utilisateurs construisant une LFS pour la première fois sont encouragés à construire sans optimisations personnalisées. Le système sera toujours très rapide et restera stable en même temps.

L'ordre dans lequel les paquets sont installés dans ce chapitre a besoin d'être strictement suivi pour s'assurer qu'aucun programme n'acquiert accidentellement un chemin ayant comme référence `/tools` en dur. Pour la même raison, ne compilez pas 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.

Avant les instructions d'installation, chaque page d'installation fournit des informations sur le paquet, incluant une description concise de ce qu'il contient, approximativement combien de temps prendra la construction et les autres paquets nécessaires lors de cette étape de construction. Suivant les instructions d'installation, il existe une liste de programmes et de bibliothèques (avec quelques brèves descriptions de ceux-ci) que le paquet installe.

Pour garder trace du paquet qui installe certains fichiers spécifiques, un gestionnaire de paquet peut être utilisé. Pour un aperçu général des différents styles de gestionnaires de paquets, merci de vous référer à <http://www.linuxfromscratch.org/blfs/view/svn/introduction/important.html>. Pour une méthode de gestion des paquets spécifiquement tournée vers LFS, nous recommandons [http://www.linuxfromscratch.org/hints/downloads/files/more\\_control\\_and\\_pkg\\_man.txt](http://www.linuxfromscratch.org/hints/downloads/files/more_control_and_pkg_man.txt).



### Note

Le reste de ce livre est à réaliser en étant connecté en tant qu'utilisateur *root* et non plus en tant qu'utilisateur *lfs*. De plus, vérifiez de nouveau que la variable `$LFS` est bien configurée.

## 6.2. Monter les systèmes de fichiers virtuels du noyau

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 $LFS/{proc,sys}
```

Maintenant, montez les systèmes de fichiers :

```
mount -vt proc proc $LFS/proc  
mount -vt sysfs sysfs $LFS/sys
```

Rappelez-vous que si vous stoppez le système LFS et que vous le relancez, il est important de vérifier que ces systèmes de fichiers sont montés avant d'entrer dans l'environnement chroot.

Des systèmes de fichiers supplémentaires seront bientôt montés à l'intérieur de l'environnement chroot. Pour garder l'hôte à jour, réalisez un « faux montage » pour chacun d'entre eux maintenant :

```
mount -vft tmpfs tmpfs $LFS/dev  
mount -vft tmpfs tmpfs $LFS/dev/shm  
mount -vft devpts -o gid=4,mode=620 devpts $LFS/dev/pts
```

## 6.3. Entrer dans l'environnement chroot

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

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

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

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

Notez 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 utilisée. Ceci survient quand le shell ne se rappelle plus les 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 sont lancés à l'intérieur de l'environnement chroot. Si vous devez quitter cet environnement quelqu'en soit la raison (un redémarrage par exemple), vous devez vous rappeler de commencer par monter les systèmes de fichiers `proc` et `devpts` (discutés dans la section précédente) et d'entrer de nouveau dans chroot avant de continuer les installations.

Notez que l'invite de **bash** dira « I have no name! » (Je n'ai pas de nom !). C'est normal car le fichier `/etc/passwd` n'existe pas encore.

## 6.4. Changer de propriétaire

Actuellement, le répertoire `/tools` appartient à l'utilisateur `lfs`. Néanmoins, ce compte utilisateur existe seulement sur votre système hôte. Bien que le répertoire `/tools` peut être supprimé une fois le système LFS terminé, il peut être conservé pour construire d'autres systèmes LFS. Si ce répertoire est conservé tel qu'il est, les fichiers appartiennent à un identifieur sans compte correspondant. Ceci est dangereux car par la suite un compte utilisateur pourrait obtenir cet identifieur et devenir soudainement le propriétaire du répertoire `/tools` et les fichiers qu'il contient, les exposant à une manipulation détournée possible.

Pour éviter ce problème, ajoutez l'utilisateur `lfs` dans votre nouveau système LFS en créant plus tard le fichier `/etc/passwd`, et en prenant garde d'affecter le bon identifieur utilisateur et groupe. Sinon, affectez le contenu du répertoire `/tools` à l'utilisateur `root` en lançant la commande suivante :

```
chown -R 0:0 /tools
```

La commande utilise `0:0` au lieu de `root:root` car **chown** n'est pas capable de résoudre le nom « root » tant que le fichier des mots de passe n'est pas créé. Ce livre suppose que vous avez exécuté la commande **chown**.

## 6.5. Créer les répertoires

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

```
install -dv /{bin,boot,dev,etc,opt,home,lib,mnt}
install -dv /{sbin,srv,usr/local,var,opt}
install -dv /root -m 0750
install -dv /tmp /var/tmp -m 1777
install -dv /media/{floppy,cdrom}
install -dv /usr/{bin,include,lib,sbin,share,src}
ln -sv share/{man,doc,info} /usr
install -dv /usr/share/{doc,info,locale,man}
install -dv /usr/share/{misc,terminfo,zoneinfo}
install -dv /usr/share/man/man{1,2,3,4,5,6,7,8}
install -dv /usr/local/{bin,etc,include,lib,sbin,share,src}
ln -sv share/{man,doc,info} /usr/local
install -dv /usr/local/share/{doc,info,locale,man}
install -dv /usr/local/share/{misc,terminfo,zoneinfo}
install -dv /usr/local/share/man/man{1,2,3,4,5,6,7,8}
install -dv /var/{lock,log,mail,run,spool}
install -dv /var/{opt,cache,lib/{misc,locate},local}
install -dv /opt/{bin,doc,include,info}
install -dv /opt/{lib,man/man{1,2,3,4,5,6,7,8}}
```

Par défaut, les répertoires sont créés avec les droits 755, ce qui n'est pas souhaitable pour tous les répertoires. Dans la commande ci-dessus, deux modifications seront effectuées : un pour le répertoire principal de *root* et un 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 le plus haut dans le masque 1777.

### 6.5.1. Remarques à propos de la conformité FHS

L'arborescence de répertoires est basée sur le standard FHS (disponible sur <http://www.pathname.com/fhs/>). En plus de cette arborescence, ce standard stipule l'existence de */usr/local/games* et */usr/share/games*. Le FHS n'est pas précis en ce qui concerne la structure du sous-répertoire */usr/local/share*, donc nous créons seulement les répertoires nécessaires. Néanmoins, n'hésitez pas à créer ces répertoires si vous préférez vous conformer plus strictement au FHS.

## 6.6. Créer les liens symboliques essentiels

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

```
ln -sv /tools/bin/{bash,cat,pwd,stty} /bin
ln -sv /tools/bin/perl /usr/bin
ln -sv /tools/lib/libgcc_s.so{,.1} /usr/lib
ln -sv bash /bin/sh
```



## 6.7. Créer les fichiers passwd, group et les journaux de trace

Pour que *root* puisse se connecter et que le nom « root » soit reconnu, il doit exister des entrées adéquates dans les fichiers `/etc/passwd` et `/etc/group`.

Créez le fichier `/etc/passwd` en lançant les commandes suivantes :

```
cat > /etc/passwd << "EOF"
root:x:0:0:root:/root:/bin/bash
EOF
```

Le mot de passe pour *root* (le caractère « x » ici est seulement pour conserver l'emplacement) sera initialisé plus tard.

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
```

Les groupes créés ne font pas partie d'un standard—they ont été choisis en partie suite aux pré-requis de la configuration d'Udev dans ce chapitre et en partie par une convention commune employée par un certain nombre de distributions Linux existantes. LSB (Linux Standard Base, disponible sur <http://www.linuxbase.org>) recommande seulement que, en plus du groupe « root » disposant d'un GID 0, un groupe « bin » de GID 1 soit présent. Tous les autres noms de groupe et GID associés sont choisis librement par l'administrateur du système car les programmes bien écrits ne dépendent pas des numéros de GID mais utilisent plutôt le nom du groupe.

Pour supprimer l'invite « I have no name! », commencez un nouveau shell. Comme un Glibc complet a été installé dans le Chapitre 5 et que les fichiers `/etc/passwd` et `/etc/group` ont été créés, la résolution des noms d'utilisateur et des noms de groupe devraient fonctionner.

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

Notez l'utilisation de la directive `+h`. Ceci indique à **bash** de ne pas utiliser son hachage interne des chemins. Sans cette directive, **bash** se rappellerait le chemin vers les binaires qu'il a exécuté. Pour utiliser les binaires dès leur installation, l'option `+h` sera utilisée pour toute la durée de ce chapitre.

Les programmes **login**, **agetty** et **init** (ainsi que d'autres) utilisent un certain nombre de journaux pour enregistrer les informations comme la personne connectée au système et sa date d'entrée. Néanmoins, ces

programmes n'écrivent pas les journaux de trace s'ils n'existent pas déjà. Initialisez les journaux 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
```

Le fichier `/var/run/utmp` enregistre les utilisateurs actuellement connectés. Le fichier `/var/log/wtmp` enregistre toutes les connexions et déconnexions. Le fichier `/var/log/lastlog` enregistre pour chaque utilisateur quand il ou elle s'est déjà connecté. Le fichier `/var/log/btmp` enregistre les tentatives échouées de connexion.

## 6.8. Peupler /dev

### 6.8.1. Créer les nœuds périphériques initiaux

Lorsque le noyau démarre le système, il requiert la présence de quelques nœuds périphériques, en particulier les périphériques `console` et `null`. Les nœuds de périphérique seront créés sur le disque dur pour être disponibles avant l'exécution de `udev`. Ils sont aussi créés quand Linux est démarré en mode simple utilisateur (d'où les droits restreints sur `console`). Créez les périphériques en exécutant les commandes suivantes :

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

### 6.8.2. Monter tmpfs et peupler /dev

La méthode recommandée pour peupler le répertoire `/dev` de périphériques est de monter un système de fichiers virtuel (comme `tmpfs`) sur le répertoire `/dev`, et d'autoriser la création dynamique des périphériques sur le système de fichiers virtuels une fois qu'ils sont détectés ou que quelque chose tente d'y accéder. Ceci est fait généralement lors du démarrage. Comme ce nouveau système n'a pas encore été démarré, il est nécessaire de faire ce que le paquetage LFS-Bootscripts aurait fait en montant `/dev` :

```
mount -nvt tmpfs none /dev
```

Le paquetage Udev est celui qui va créer les périphériques dans le répertoire `/dev`. Comme il ne sera installé que plus tard lors de ce processus, créez manuellement l'ensemble minimum de nœuds de périphériques nécessaire pour terminer la construction de ce système :

```
mknod -m 622 /dev/console c 5 1
mknod -m 666 /dev/null c 1 3
mknod -m 666 /dev/zero c 1 5
mknod -m 666 /dev/ptmx c 5 2
mknod -m 666 /dev/tty c 5 0
mknod -m 444 /dev/random c 1 8
mknod -m 444 /dev/urandom c 1 9
chown -v root:tty /dev/{console,ptmx,TTY}
```

Certains liens symboliques et répertoires sont requis par LFS qui les crée lors du démarrage du système grâce au paquetage LFS-Bootscripts. Comme il s'agit d'un environnement chroot et que nous n'avons pas démarré avec lui, ces liens symboliques et répertoires doivent être créés ici :

```
ln -sv /proc/self/fd /dev/fd
ln -sv /proc/self/fd/0 /dev/stdin
ln -sv /proc/self/fd/1 /dev/stdout
ln -sv /proc/self/fd/2 /dev/stderr
ln -sv /proc/kcore /dev/core
mkdir -v /dev/pts
mkdir -v /dev/shm
```

Enfin, montez les bons systèmes de fichiers virtuels (noyau) sur les répertoires nouvellement créés:

```
mount -vt devpts -o gid=4,mode=620 none /dev/pts
mount -vt tmpfs none /dev/shm
```

Les commandes **mount** exécutées ci-dessus pourraient causer les messages d'avertissement suivants :

```
can't open /etc/fstab: No such file or directory.
```

Ce fichier—`/etc/fstab`—n'a pas été encore créé mais il n'est pas non plus requis pour le bon montage des systèmes de fichiers. De cette façon, le message peut être ignoré sans crainte.

## 6.9. Linux-Libc-Headers-2.6.11.2

Le paquet Linux-Libc-Headers contient les en-têtes du noyau « nettoyés ».

**Temps de construction estimé :** 0,1 SBU

**Espace disque requis :** 26,9 Mo

**Dépendances de l'installation :** Coreutils

### 6.9.1. Installation de Linux-Libc-Headers

Pendant des années, une pratique courante était d'utiliser les en-têtes « bruts » du noyau (provenant directement de l'archive tar du noyau) dans `/usr/include`, mais sur les quelques années précédentes, les développeurs du noyau ont acquis la conviction que cela ne devait pas se passer ainsi. Cela a donné naissance au projet Linux-Libc-Headers, qui a été conçu pour maintenir une version stable de l'API des en-têtes de Linux.

Installez les fichiers d'en-tête :

```
cp -Rv include/asm-i386 /usr/include/asm
cp -Rv include/linux /usr/include
```

Assurez-vous que tous les en-têtes appartiennent bien à root :

```
chown -Rv root:root /usr/include/{asm,linux}
```

Assurez-vous que tous les utilisateurs puissent lire les en-têtes :

```
find /usr/include/{asm,linux} -type d -exec chmod -v 755 {} \;
find /usr/include/{asm,linux} -type f -exec chmod -v 644 {} \;
```

### 6.9.2. Contenu de Linux-Libc-Headers

**En-têtes installés:** `/usr/include/{asm,linux}/*.h`

#### Descriptions courtes

`/usr/include/{asm,linux}/*.h` Les en-têtes de l'API de Linux

## 6.10. Man-pages-2.01

Le paquet Man-pages contient plus de 1 200 pages man.

**Temps de construction estimé :** 0,1 SBU

**Espace disque requis :** 25,8 Mo

**Dépendances de l'installation :** Bash, Coreutils et Make

### 6.10.1. Installation de Man-pages

Installez Man-pages en lançant :

```
make install
```

### 6.10.2. Contenu de Man-pages

**Fichiers installés:** plusieurs pages man

#### Descriptions courtes

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

## 6.11. Glibc-2.3.4

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

**Temps de construction estimé :** 12,3 SBU

**Espace disque requis :** 476 Mo

**Dépendances de l'installation :** Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Gettext, Grep, Make, Perl, Sed et Texinfo

### 6.11.1. Installation de Glibc



#### Note

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

Ce paquet est connu pour avoir des soucis quand les options d'optimisation par défaut (en incluant les options `-march` et `-mcpu`) sont modifiées. Donc, si des variables d'environnement qui surchargent les optimisations par défaut, telles que `CFLAGS` et `CXXFLAGS`, ont été définies, supprimez cette initialisation pour la construction de GCC.

Le système de construction de la Glibc est très bien fait et s'installera 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 Glibc parce que les tests d'autoconf de Glibc donneraient alors des résultats faussés, défaussant ainsi notre but d'achever une construction propre.

L'archive tar linuxthreads contient les pages man pour les bibliothèques de threading installées par Glibc. Déballez l'archive tar à l'intérieur du répertoire source Glibc :

```
tar -xjvf ../glibc-linuxthreads-2.3.4.tar.bz2
```

Dans de rares circonstances, Glibc peut générer une erreur de segmentation quand aucun répertoire de recherche standard n'existe. Le correctif suivant s'occupe de ce problème :

```
patch -Np1 -i ../glibc-2.3.4-rtld_search_dirs-1.patch
```

Glibc contient deux tests qui échoueront si le noyau en cours d'exécution est un 2.6.11.x. Le problème se situe sur les tests eux-même, pas avec la libc ou le noyau. Ce correctif corrige le problème :

```
patch -Np1 -i ../glibc-2.3.4-fix_test-1.patch
```

Appliquez le correctif suivant pour corriger un bogue dans Glibc qui peut empêcher certains programmes (comme OpenOffice.org) de fonctionner correctement :

```
patch -Np1 -i ../glibc-2.3.4-tls_assert-1.patch
```

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

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

Préparez la compilation de Glibc :

```
../glibc-2.3.4/configure --prefix=/usr \
  --disable-profile --enable-add-ons \
  --enable-kernel=2.6.0 --libexecdir=/usr/lib/glibc
```

Voici la signification des options de configuration :

```
--libexecdir=/usr/lib/glibc
```

Ceci modifie l'emplacement du programme `pt_chown`, dont la valeur par défaut est `/usr/libexec`, par `/usr/lib/glibc`.

Compilez le paquet :

```
make
```



### Important

Dans cette section, la suite de tests de Glibc est considérée comme critique. Ne pas la laisser passer quelque soient les circonstances.

Testez les résultats :

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

La suite de tests Glibc est grandement dépendante de certaines fonctions de l'hôte système, en particulier le noyau. En général, la suite de tests Glibc devrait toujours réussir. Néanmoins, dans certaines circonstances, quelques échecs sont inévitables. Voici une liste des problèmes les plus fréquents :

- Les tests *math* échouent quelque fois lors de leur exécution sur des systèmes où le processeur n'est pas un Intel ou un AMD authentique. Certains paramètres d'optimisation sont aussi un facteur connu pour ce type de problèmes.
- Les tests *gettext* échouent quelque fois à cause de problèmes sur le système hôte, les raisons exactes n'étant pas encore claires.
- Si vous avez monté la partition LFS 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 LFS.
- Lors d'une exécution sur un matériel ancien et lent, quelques tests peuvent échouer à cause de délais excédés.

Bien que ce ne soit qu'un simple message, l'étape d'installation de Glibc se plaindra de l'absence de `/etc/ld.so.conf`. Supprimez ce message d'avertissement avec :

```
touch /etc/ld.so.conf
```

Installez le paquet :

```
make install
```



Les locales qui permettent à votre système de répondre en une langue différente n'ont pas été installées avec la commande ci-dessus. Installez-les avec ceci :

```
make localedata/install-locales
```

Pour gagner du temps, une alternative à la commande précédente (qui génère et installe toutes les locales qu'il trouve dans le fichier `glibc-2.3.4/localedata/SUPPORTED`) est d'installer uniquement les locales que vous souhaitez et dont vous avez besoin. Ceci se fait en utilisant la commande **localedef**. Des informations sur cette commande sont disponibles dans le fichier `INSTALL` des sources de Glibc. Néanmoins, il existe un certain nombre de locales essentielles pour réussir les tests des paquets futurs, en particulier les tests de *libstdc++*. Les instructions suivantes, contrairement à la cible *install-locales* ci-dessus, installeront l'ensemble minimal des locales nécessaires pour que les tests se passent dans de bonnes conditions :

```
mkdir -pv /usr/lib/locale
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 comprises dans les livres LFS et BLFS. À cause des différents problèmes survenus parce que les développeurs des applications ont fait des choix qui ont cassé ces locales, LFS ne devrait pas être utilisé avec des locales qui utilisent des ensembles de caractères à plusieurs octets (ceci incluant UTF-8) ou l'ordre d'écriture de droite à gauche. De nombreux correctifs officiels et instables sont requis pour corriger ces problèmes et il a été décidé par les développeurs de LFS que ces locales complexes ne seraient pas supportées en ce moment. Ceci s'applique aussi aux locales `ja_JP` et `fa_IR`—elles ont été installées seulement pour que les tests de GCC et Gettext réussissent bien que le programme **watch** (un composant du paquetage Procps) ne fonctionne pas correctement avec elles. Différents essais pour contourner ces restrictions sont documentés dans les astuces relatives à l'internationalisation.

Construisez les pages man de `linuxthreads` qui sont une grande référence à l'API des threads (applicable aussi à NPTL) :

```
make -C ../glibc-2.3.4/linuxthreads/man
```

Installez ces pages :

```
make -C ../glibc-2.3.4/linuxthreads/man install
```

## 6.11.2. Configurer Glibc

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

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

```
cat > /etc/nsswitch.conf << "EOF"
# Début /etc/nsswitch.conf

passwd: files
group: files
shadow: files

hosts: files dns
networks: files

protocols: files
services: files
ethers: files
rpc: files

# Fin /etc/nsswitch.conf
EOF
```

Pour déterminer dans quel fuseau horaire vous vous situez, lancez le script suivant :

```
tzselect
```

Lorsque avoir répondu à quelques questions sur votre emplacement, le script affichera le nom du fuseau horaire (quelque chose comme *EST5EDT* ou *Canada/Eastern*). Ensuite, créez le fichier `/etc/localtime` en lançant :

```
cp -v --remove-destination /usr/share/zoneinfo/[xxx] \
/etc/localtime
```

Remplacez `[xxx]` avec le nom du fuseau horaire que **tzselect** a fourni (c'est-à-dire *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 de se couvrir de la situation où `/usr` serait une partition séparée. Ceci pourrait arriver, par exemple, en démarrant en mode simple utilisateur.

### 6.11.3. Configurer le chargeur dynamique

Par défaut, le chargeur dynamique (`/lib/ld-linux.so.2`) cherche les bibliothèques partagées, nécessaires aux programmes lors de leur exécution, dans `/lib` et `/usr/lib`. Néanmoins, s'il existe des bibliothèques dans d'autres répertoires que `/lib` et `/usr/lib`, leur emplacement doit être ajouté dans le fichier `/etc/ld.so.conf` pour que le chargeur dynamique les trouve. `/usr/local/lib` et `/opt/lib` sont deux répertoires connus pour contenir des bibliothèques supplémentaires, donc ajoutez ces deux répertoires au chemin de recherche du chargeur dynamique.

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

```
cat > /etc/ld.so.conf << "EOF"
# Début /etc/ld.so.conf

/usr/local/lib
/opt/lib

# Fin /etc/ld.so.conf
EOF
```

### 6.11.4. Contenu de Glibc

**Programmes installés:** `catchsegv`, `gencat`, `getconf`, `getent`, `iconv`, `iconvconfig`, `ldconfig`, `ldd`, `lddlibc4`, `locale`, `localedef`, `mtrace`, `nscd`, `nscd_nischeck`, `pcprofiledump`, `pt_chown`, `rpcgen`, `rpcinfo`, `sln`, `sprof`, `tzselect`, `xtrace`, `zdump` et `zic`

**Bibliothèques installées:** `ld.so`, `libBrokenLocale.[a,so]`, `libSegFault.so`, `libanl.[a,so]`, `libbsd-compat.a`, `libc.[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]`, `libresolv.[a,so]`, `librpcsvc.a`, `librt.[a,so]`, `libthread_db.so` et `libutil.[a,so]`

#### Descriptions courtes

<b>catchsegv</b>	Peut être utilisé pour créer une trace de la pile lorsqu'un programme s'arrête avec une erreur de segmentation
<b>gencat</b>	Génère des catalogues de messages
<b>getconf</b>	Affiche les valeurs de configuration du système pour les variables spécifiques du système de fichiers
<b>getent</b>	Récupère les entrées à partir d'une base de données administrative
<b>iconv</b>	Réalise une conversion de l'ensemble des caractères
<b>iconvconfig</b>	Crée un fichier de configuration pour le module <b>iconv</b> fastloading
<b>ldconfig</b>	Configure les liens du chargeur dynamique
<b>ldd</b>	Indique les bibliothèques partagées requises pour chaque programme ou bibliothèque partagée
<b>lddlibc4</b>	Assiste <b>ldd</b> avec des fichiers objets
<b>locale</b>	Indique au compilateur d'activer ou de désactiver l'utilisation des locales POSIX pour les opérations internes
<b>localedef</b>	Compile les spécifications des locales

<b>mtrace</b>	Lit et interprète un fichier de trace mémoire et affiche un résumé dans un format lisible par un humain
<b>nscd</b>	Un démon pour les services de noms fournissant un cache pour les requêtes les plus communes
<b>nscd_nischeck</b>	Vérifie si le mode sécurisé est nécessaire pour les recherches NIS+
<b>pcprofiledump</b>	Affiche des informations générées par un profilage du PC
<b>pt_chown</b>	un programme d'aide pour que <b>grantpt</b> initialise les droits des propriétaires, groupes et autres d'un pseudo-terminal esclave
<b>rpcgen</b>	Génère du code C pour implémenter le protocole RPC ( <i>Remote Procedure Call</i> )
<b>rpcinfo</b>	Fait un appel RPC à un serveur RPC
<b>sln</b>	Un programme <b>ln</b> lié statiquement
<b>sprof</b>	Lit et affiche les données de profilage des objets partagés
<b>tzselect</b>	Demande à l'utilisateur l'emplacement géographique du système et donne la description du fuseau horaire correspondante
<b>xtrace</b>	Trace l'exécution d'un programme en affichant la fonction en cours d'exécution
<b>zdump</b>	Afficheur de fuseau horaire
<b>zic</b>	Compilateur de fuseau horaire
ld.so	Le programme d'aide des bibliothèques partagées exécutables
libBrokenLocale	Utilisé par des programmes comme Mozilla pour résoudre les locales cassées
libSegFault	Un gestionnaire de signaux d'erreurs de segmentation
libanl	Une bibliothèque asynchrone de recherche de noms
libbsd-compat	Fournit la portabilité nécessaire pour faire fonctionner certains programmes BSD sous Linux
libc	La principale bibliothèque C
libcrypt	La bibliothèque de cryptographie
libdl	La bibliothèque de l'interface du chargeur dynamique
libg	Une bibliothèque d'exécution pour g++
libieee	La bibliothèque des nombres flottants IEEE ( <i>Institute of Electrical and Electronic Engineers</i> )
libm	La bibliothèque mathématique
libmcheck	Contient du code à lancer au démarrage
libmemusage	Utilisé par <b>memusage</b> pour collecter des informations sur l'utilisation mémoire d'un programme
libnsl	La bibliothèque de services réseau
libnss	Les bibliothèques « Name Service Switch », contenant des fonctions de résolution de noms d'hôtes, de noms d'utilisateurs, de noms de groupes, d'alias, de services, de protocoles et ainsi de suite

<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 « standard » utilisées par de nombreux outils Unix

## 6.12. Ré-ajustement de l'ensemble d'outils

Maintenant que les bibliothèques C finales ont été installées, il est temps d'ajuster de nouveau l'ensemble d'outils. L'ensemble d'outils sera ajusté de façon à ce qu'il lie tout programme nouvellement compilé avec ces nouvelles bibliothèques. C'est le même processus que celui utilisé dans la phase « Ajustement » au début du Chapitre 5, avec les ajustements inversés. Dans Chapitre 5, l'ensemble était passé des répertoires `/usr/lib` de l'hôte dans le nouveau répertoire `/tools/lib`. Maintenant, l'ensemble sera guidé du même répertoire `/tools/lib` vers les répertoires `/usr/lib` de LFS.

Commencez en ajustant l'éditeur de liens. Les répertoires des sources et de construction de la deuxième passe de Binutils ont été conservés dans ce but. Installez l'éditeur de liens ajusté en exécutant la commande suivante à partir du répertoire `binutils-build` :

```
make -C ld INSTALL=/tools/bin/install install
```



### Note

Si le précédent avertissement pour conserver les répertoires des sources et de construction de Binutils lors de la deuxième passe dans Chapitre 5 a été oublié, ou s'ils ont été accidentellement supprimés ou rendus inaccessibles, ignorez la commande ci-dessus. Le résultat en sera que le prochain paquet, Binutils, sera lié aux bibliothèques C dans `/tools` plutôt que dans `/usr/lib`. Ceci n'est pas idéal. Néanmoins, les tests ont montrés que les binaires Binutils résultants devraient être identiques.

À partir de maintenant, chaque programme compilé sera lié avec les bibliothèques de `/usr/lib` et de `/lib`. L'option supplémentaire `INSTALL=/tools/bin/install` est nécessaire car le fichier Makefile créé lors de la seconde passe contient toujours la référence à `/usr/bin/install`, qui n'a pas encore été installé. Quelques distributions hôtes contiennent un lien symbolique `ginstall` qui est prioritaire dans le fichier Makefile et peut cause un problème. La commande ci-dessus tient compte de ce problème.

Supprimez les répertoires des sources et de construction de Binutils maintenant.

Ensuite, modifiez le fichier specs de GCC pour qu'il pointe sur le nouvel éditeur de liens. Une commande `perl` accomplit ceci :

```
perl -pi -e 's@ /tools/lib/ld-linux.so.2@ /lib/ld-linux.so.2@g;' \
-e 's@*startfile_prefix_spec:\n@$_/usr/lib/ @g;' \
`gcc --print-file specs`
```

C'est une bonne idée d'inspecter visuellement le fichier specs pour vérifier que les modifications attendues ont réellement été effectuées.



### Important

En travaillant sur une plateforme où le nom de l'éditeur de liens est quelque chose d'autres que `ld-linux.so.2`, substituez « `ld-linux.so.2` » par le nom de l'éditeur de liens dans les commandes suivantes. Référez-vous à Section 5.2, « Notes techniques sur l'ensemble d'outils, » si nécessaire.



## 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 attendu. Pour cela, réalisez une petite vérification :

```
echo 'main(){}' > dummy.c
cc 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]
```

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

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

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

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

## 6.13. Binutils-2.15.94.0.2.2

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

**Temps de construction estimé :** 1,3 SBU

**Espace disque requis :** 158 Mo

**Dépendances de l'installation :** Bash, Bison, Coreutils, Diffutils, Flex, GCC, Gettext, Glibc, Grep, M4, Make, Perl, Sed et Texinfo

### 6.13.1. Installation de Binutils

Ce paquet est connu pour avoir des soucis quand les options d'optimisation par défaut (en incluant les options `-march` et `-mcpu`) sont modifiées. Donc, si des variables d'environnement qui surchargent les optimisations par défaut, telles que `CFLAGS` et `CXXFLAGS`, ont été définies, supprimez cette initialisation pour la construction de Binutils.

Vérifiez que les pseudo-terminaux (PTY) fonctionnent correctement dans l'environnement chroot. Vérifiez que tout est bien configuré en effectuant un test simple :

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

Si le message suivant apparaît, l'environnement chroot n'est pas configuré correctement pour des opérations sur les PTY :

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

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 :

```
../binutils-2.15.94.0.2.2/configure --prefix=/usr \
--enable-shared
```

Compilez le paquet :

```
make tooldir=/usr
```

Normalement, le répertoire `tooldir` (celui où seront placés les exécutable) est configuré avec `$(exec_prefix)/$(target_alias)`. Par exemple, les machines `i686` l'étendront en `/usr/i686-pc-linux-gnu`. Comme il s'agit d'un système personnalisé, nous n'avons pas besoin d'un répertoire spécifique à notre cible dans `/usr`. `$(exec_prefix)/$(target_alias)` serait utilisée si le système avait pour but une cross-compilation (par exemple, compiler un paquet sur une machine Intel qui génère du code pouvant être exécuté sur des machines PowerPC).



#### Important

La suite de tests de Binutils dans cette section est considérée comme critique. Ne pas la laissez



passer, quelqu'en soit la raison.

Testez les résultats :

```
make check
```

Installez le paquet :

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

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

```
cp -v ../binutils-2.15.94.0.2.2/include/libiberty.h /usr/include
```

## 6.13.2. Contenu de Binutils

**Programmes installés:** `addr2line`, `ar`, `as`, `c++filt`, `gprof`, `ld`, `nm`, `objcopy`, `objdump`, `ranlib`, `readelf`, `size`, `strings` et `strip`

**Bibliothèques installées:** `libiberty.a`, `libbfd.[a,so]` et `libopcodes.[a,so]`

### Descriptions courtes

<b>addr2line</b>	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
<b>ar</b>	Crée, modifie et extrait à partir d'archives
<b>as</b>	Un assembleur qui assemble la sortie de <b>gcc</b> en un fichier objet
<b>c++filt</b>	Utilisé par l'éditeur de liens pour récupérer les symboles C++ et Java, et pour empêcher les fonctions surchargées d'arrêter brutalement le programme
<b>gprof</b>	Affiche les données de profilage d'appels dans un graphe
<b>ld</b>	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
<b>nm</b>	Liste les symboles disponibles dans un fichier objet
<b>objcopy</b>	Traduit un type de fichier objet en un autre
<b>objdump</b>	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
<b>ranlib</b>	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
<b>readelf</b>	Affiche des informations sur les binaires du type ELF
<b>size</b>	Liste les tailles des sections et la taille totale pour les fichiers objets donnés
<b>strings</b>	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
<b>strip</b>	Supprime les symboles des fichiers objets
libiberty	contient des routines utilisées par différents programmes GNU comme <b>getopt</b> , <b>obstack</b> , <b>strerror</b> , <b>strtol</b> et <b>strtoul</b>
libbfd	La 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 <b>objdump</b> .

## 6.14. GCC-3.4.3

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

**Temps de construction estimé :** 11,7 SBU

**Espace disque requis :** 451 Mo

**Dépendances de l'installation :** Bash, Binutils, Coreutils, Diffutils, Findutils, Gawk, Gettext, Glibc, Grep, Make, Perl, Sed et Texinfo

### 6.14.1. Installation de GCC

Ce paquet est connu pour avoir des soucis quand les options d'optimisation par défaut (en incluant les options `-march` et `-mcpu`) sont modifiées. Donc, si des variables d'environnement qui surchargent les optimisations par défaut, telles que `CFLAGS` et `CXXFLAGS`, ont été définies, supprimez cette initialisation pour la construction de GCC.

Appliquez seulement le correctif No-Fixincludes (et pas Specs) utilisé aussi dans le chapitre précédent :

```
patch -Np1 -i ../gcc-3.4.3-no_fixincludes-1.patch
```

La compilation de GCC échoue pour certains paquets en dehors d'une installation LFS de base (par exemple, Mozilla et kdegraphics) si GCC est utilisé avec les dernières versions de Binutils. Appliquez le correctif suivant pour corriger ce problème :

```
patch -Np1 -i ../gcc-3.4.3-linkonce-1.patch
```

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

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

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

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

Préparez la compilation de GCC :

```
../gcc-3.4.3/configure --prefix=/usr \
  --libexecdir=/usr/lib --enable-shared \
  --enable-threads=posix --enable-__cxa_atexit \
  --enable-clocale=gnu --enable-languages=c,c++
```

Compilez le paquet :

```
make
```

**Important**

Dans cette section, la suite de tests pour GCC est considérée critique. Ne pas la laisser passer quelque soient les circonstances.

Testez les résultats mais ne vous arrêtez pas aux erreurs :

```
make -k check
```

Quelques erreurs sont connues et ont été indiquées dans le chapitre précédent. Les notes de la suite de tests sur Section 5.11, « GCC-3.4.3 - Passe 2 » sont toujours très appropriées ici. Assurez-vous de vous y référer si nécessaire.

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
```

**Note**

À ce moment, il est fortement recommandé de répéter les vérifications réalisées plus tôt dans ce chapitre. Référez-vous à Section 6.12, « Ré-ajustement de l'ensemble d'outils » et répétez la vérification. Si les résultats sont mauvais, alors il y a des chances pour que le correctif GCC Specs ait été mal appliqué à partir de Chapitre 5.

## 6.14.2. Contenu de GCC

**Programmes installés:** `c++`, `cc` (lien vers `gcc`), `cpp`, `g++`, `gcc`, `gccbug` et `gcov`

**Bibliothèques installées:** `libgcc.a`, `libgcc_eh.a`, `libgcc_s.so`, `libstdc++.a`, `libstdc++.so` et `libsupc++.a`

### Descriptions courtes

<code>cc</code>	Le compilateur C
<code>cpp</code>	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
<code>c++</code>	Le compilateur C++
<code>g++</code>	Le compilateur C++
<code>gcc</code>	Le compilateur C
<code>gccbug</code>	Un script shell utilisé pour aider à la création de bons rapports de bogues

- gcov** Un outil de tests ; il est utilisé pour analyser les programmes et savoir où des optimisations seraient suivies du plus d'effet
- libgcc** contient un support en exécution pour **gcc**
- libstdc++** La bibliothèque C++ standard
- libsupc++** Fournit des routines de support pour le langage de programmation C++

## 6.15. Coreutils-5.2.1

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

**Temps de construction estimé :** 0,9 SBU

**Espace disque requis :** 52,8 Mo

**Dépendances de l'installation :** Bash, Binutils, Coreutils, Diffutils, GCC, Gettext, Glibc, Grep, Make, Perl et Sed

### 6.15.1. 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 les architectures Intel :

```
patch -Np1 -i ../coreutils-5.2.1-uname-2.patch
```

Empêchez Coreutils d'installer des binaires qui pourraient être installés plus tard par d'autres paquets :

```
patch -Np1 -i ../coreutils-5.2.1-suppress_uptime_kill_su-1.patch
```

Maintenant, préparez la compilation de Coreutils :

```
DEFAULT_POSIX2_VERSION=199209 ./configure --prefix=/usr
```

Compilez le paquet :

```
make
```

La suite de tests de Coreutils fait quelques suppositions sur la présence d'utilisateurs et de groupes systèmes qui ne sont pas valides à l'intérieur de l'environnement minimal qui existe pour le moment. Du coup, des éléments supplémentaires doivent être configurés avant de lancer ces tests. Allez directement à « Installez le paquet » si vous ne comptez pas lancer les tests.

Créez deux groupes et un utilisateur :

```
echo "dummy1:x:1000:" >> /etc/group
echo "dummy2:x:1001:dummy" >> /etc/group
echo "dummy:x:1000:1000:::/bin/bash" >> /etc/passwd
```

Maintenant, la suite de tests peut être lancée. Tout d'abord, lancez les quelques tests qui ont besoin d'être lancé en tant que `root` :

```
make NON_ROOT_USERNAME=dummy check-root
```

Puis, lancez le reste des tests en tant qu'utilisateur `dummy` :

```
src/su dummy -c "make RUN_EXPENSIVE_TESTS=yes check"
```

Une fois les tests terminés, supprimez l'utilisateur et les groupes :

```
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,dd,df,echo} /bin
mv -v /usr/bin/{false,hostname,ln,ls,mkdir,mknod,mv,pwd,rm} /bin
mv -v /usr/bin/{rmdir,stty,sync,true,uname} /bin
mv -v /usr/bin/chroot /usr/sbin
```

Certains des scripts du paquet LFS-Bootscripts dépendent de **head** et **sleep**. Comme `/usr` pourrait ne pas être disponible dans les premières phases du démarrage, ces binaires ont besoin d'être sur la partition `root` :

```
mv -v /usr/bin/{head,sleep} /bin
```

## 6.15.2. Contenu de Coreutils

**Programmes installés:** basename, cat, 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, mv, nice, nl, nohup, od, paste, pathchk, pinky, pr, printenv, printf, ptx, pwd, readlink, rm, rmdir, seq, sha1sum, shred, sleep, sort, split, stat, stty, sum, sync, tac, tail, tee, test, touch, tr, true, tsort, tty, uname, unexpand, uniq, unlink, users, vdir, wc, who, whoami et yes

### Descriptions courtes

<b>basename</b>	Supprime tout le chemin et un suffixe donné à partir du nom de fichier donné
<b>cat</b>	Concatène des fichiers sur la sortie standard
<b>chgrp</b>	Change le groupe propriétaire de certains fichiers et répertoires.
<b>chmod</b>	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
<b>chown</b>	Modifie le propriétaire utilisateur et/ou groupe de certains fichiers et répertoires
<b>chroot</b>	Lance une commande avec le répertoire spécifié comme répertoire racine (/)
<b>cksum</b>	Affiche la somme de vérification CRC (Cyclic Redundancy Check) et le nombre d'octets de chaque fichier
<b>comm</b>	Compare deux fichiers triés, affichant sur trois colonnes, les lignes uniques et les lignes communes
<b>cp</b>	Copie des fichiers
<b>csplit</b>	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
<b>cut</b>	Affiche des parties de lignes, sélectionnant ces parties suivant des champs ou positions donnés
<b>date</b>	Affiche l'heure actuelle dans le format donné ou initialise la date système
<b>dd</b>	Copie un fichier en utilisant la taille de bloc donnée et le nombre, tout en réalisant des conversions optionnelles

<b>df</b>	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
<b>dir</b>	Liste le contenu de chaque répertoire donné (identique à la commande <b>ls</b> )
<b>dircolors</b>	Affiche les commandes pour initialiser la variable d'environnement <code>LS_COLOR</code> , ce qui permet de changer le schéma de couleurs utilisé par <b>ls</b>
<b>dirname</b>	Supprime le suffixe qui ne représente pas le répertoire dans un nom de fichier donné
<b>du</b>	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
<b>echo</b>	Affiche les chaînes données
<b>env</b>	Lance une commande dans un environnement modifié
<b>expand</b>	Convertit les tabulations en espaces
<b>expr</b>	Évalue des expressions
<b>factor</b>	Affiche les facteurs premiers de tous les entiers spécifiés
<b>false</b>	Ne fait rien. Il renvoie toujours un un code d'erreur indiquant l'échec
<b>fmt</b>	Reformate les paragraphes dans les fichiers donnés
<b>fold</b>	Emballer les lignes des fichiers donnés
<b>groups</b>	Affiche les groupes auxquels appartient un utilisateur
<b>head</b>	Affiche les dix premières lignes (ou le nombre demandé de lignes) pour chaque fichier précisé
<b>hostid</b>	Affiche l'identifiant numérique de l'hôte en hexadécimal
<b>hostname</b>	Affiche ou initialise le nom de l'hôte
<b>id</b>	Affiche l'identifiant effectif de l'utilisateur courant ou de l'utilisateur précisé, l'identifiant du groupe et les groupes auxquels appartient cet utilisateur
<b>install</b>	Copie les fichiers en initialisant leur droits et, si possible, leur propriétaire et groupe
<b>join</b>	Joint à partir de deux fichiers les lignes qui ont des champs de jointure identiques
<b>link</b>	Crée un lien physique avec le nom de donné vers le fichier indiqué
<b>ln</b>	Crée des liens symboliques ou physiques entre des fichiers
<b>logname</b>	Indique le nom de connexion de l'utilisateur actuel
<b>ls</b>	Liste le contenu de chaque répertoire donné
<b>md5sum</b>	Affiche ou vérifie les sommes de vérification MD5 (Message Digest 5)
<b>mkdir</b>	Crée des répertoires avec les noms donnés
<b>mkfifo</b>	Crée des fichiers FIFOs (First-In, First-Out, un « tube nommé » dans le vocabulaire d'Unix) avec les noms donnés
<b>mknod</b>	Crée des noeuds périphérique avec les noms donnés. Un noeud périphérique est un fichier spécial de type caractère ou bloc, ou encore un FIFO
<b>mv</b>	Déplace ou renomme des fichiers ou répertoires



<b>nice</b>	Lance un programme avec un priorité modifiée
<b>nl</b>	Numérote les lignes des fichiers donnés
<b>nohup</b>	Lance une commande immune aux arrêts brutaux, dont la sortie est redirigée vers un journal de traces
<b>od</b>	Affiche les fichiers en octal ou sous d'autres formes
<b>paste</b>	Joint les fichiers donnés en faisant plaçant les lignes correspondantes ligne par ligne en les séparant par des caractères de tabulation
<b>pathchk</b>	Vérifie que les noms de fichier sont valides ou portables
<b>pinky</b>	Un client « finger » léger. Il affiche quelques informations sur les utilisateurs indiqués
<b>pr</b>	Fait de la pagination, principalement en colonne, des fichiers pour une impression
<b>printenv</b>	Affiche l'environnement
<b>printf</b>	Affiche les arguments donnés suivant le format demandé, un peu comme la fonction C printf
<b>ptx</b>	Produit un index permuté à partir du contenu des fichiers indiqués, avec le contexte de chaque mot
<b>pwd</b>	Indique le nom du répertoire courant
<b>readlink</b>	Indique la valeur du lien symbolique
<b>rm</b>	Supprime des fichiers ou des répertoires
<b>rmdir</b>	Supprime des répertoires s'ils sont vides
<b>seq</b>	Affiche une séquence de nombres, à l'intérieur d'une échelle et avec un incrément spécifié
<b>sha1sum</b>	Affiche ou vérifie des sommes de vérification 160-bit SHA1
<b>shred</b>	Efface les fichiers indiqués en écrivant dessus des modèles aléatoires pour rendre la récupération des données très difficile
<b>sleep</b>	Fait une pause d'un certain temps
<b>sort</b>	Trie les lignes des fichiers donnés
<b>split</b>	Divise les fichiers donnés en plusieurs pièces, par taille ou par nombre de lignes
<b>stat</b>	Affiche le statut du fichier ou du système de fichiers
<b>stty</b>	Initialise ou affiche les paramètres de la ligne du terminal
<b>sum</b>	Affiche la somme de vérification et le nombre de blocs pour chacun des fichiers donnés
<b>sync</b>	Vide les tampons du système de fichiers. Cela force l'enregistrement des blocs modifiés sur disque et met à jour le superbloc
<b>tac</b>	Concatène les fichiers donnés à l'envers
<b>tail</b>	Affiche les dix dernières lignes (ou le nombre de lignes indiqué) pour chaque fichier précisé
<b>tee</b>	Lit à partir de l'entrée standard en écrivant à la fois sur la sortie standard et sur les fichiers indiqués

<b>test</b>	Compare les valeurs et vérifie les types de fichiers
<b>touch</b>	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 inexistantes sont créés avec une longueur nulle
<b>tr</b>	Traduit, réduit et supprime les caractères donnés à partir de l'entrée standard
<b>true</b>	Ne fait rien mais avec succès. Il quitte avec un code de sortie indiquant une réussite
<b>tsort</b>	Réalise un tri topologique. Il écrit une liste totalement ordonnée suivant un fichier donné partiellement ordonné
<b>tty</b>	Indique le nom du fichier du terminal connecté à l'entrée standard
<b>uname</b>	Affiche les informations système
<b>unexpand</b>	Convertit les espaces en tabulations
<b>uniq</b>	Conserve qu'une ligne sur plusieurs lignes identiques successivement
<b>unlink</b>	Supprime le fichier donné
<b>users</b>	Indique les noms des utilisateurs actuellement connectés
<b>vdir</b>	Est identique à <b>ls -l</b>
<b>wc</b>	Indique le nombre de lignes, mots et octets de chaque fichier indiqué ainsi que le nombre total de ligne lorsque plus d'un fichier est donné
<b>who</b>	Indique qui est connecté
<b>whoami</b>	Indique le nom de l'utilisateur associé avec l'identifiant utilisateur effectif
<b>yes</b>	Affiche « y » ou la chaîne précisée de manière répétée jusqu'à être tué

## 6.16. Zlib-1.2.3

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

**Temps de construction estimé :** 0,1 SBU

**Espace disque requis :** 3,1 Mo

**Dépendances de l'installation :** Binutils, Coreutils, GCC, Glibc, Make et Sed

### 6.16.1. Installation de Zlib



#### Note

Zlib est connu pour mal construire sa bibliothèque partagée si `CFLAGS` fait partie de l'environnement. En initialisant une variable `CFLAGS`, assurez-vous d'ajouter la directive `-fPIC` à la variable `CFLAGS` pour la durée de la commande **configure** ci-dessous puis de la supprimer après coup.

Préparez la compilation de Zlib :

```
./configure --prefix=/usr --shared --libdir=/lib
```

Compilez le paquet :

```
make
```

Pour tester les résultats, lancez : **make check**.

Installez la bibliothèque partagée :

```
make install
```

La commande précédente a installé un fichier `.so` dans `/lib`. Nous le supprimerons et créerons de nouveau un lien vers `/usr/lib`:

```
rm -v /lib/libz.so
ln -sfv ../../lib/libz.so.1.2.3 /usr/lib/libz.so
```

Construisez aussi la bibliothèque statique :

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

Pour tester de nouveau les résultats, lancez : **make check**.

Installez la bibliothèque statique :

```
make install
```

Corrigez les droits sur la bibliothèque statique :

```
chmod -v 644 /usr/lib/libz.a
```

### 6.16.2. Contenu de Zlib

**Bibliothèques installées:** libz.[a,so]

## **Descriptions courtes**

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

## 6.17. Mktemp-1.5

Le paquet Mktemp contient des programmes utilisés pour créer des fichiers temporaires sécurisés à partir de scripts shell.

**Temps de construction estimé :** 0,1 SBU

**Espace disque requis :** 436 Ko

**Dépendances de l'installation :** Coreutils, Make et Patch

### 6.17.1. Installation de Mktemp

Beaucoup de scripts utilisent toujours le programme obsolète **tempfile**, qui dispose à peu près des mêmes fonctionnalités que **mktemp**. Corrigez mktemp pour inclure un emballage **tempfile** :

```
patch -Np1 -i ../mktemp-1.5-add_tempfile-2.patch
```

Préparez la compilation de Mktemp :

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

Voici la signification de l'option de configure :

*--with-libc*

Ceci fait que le programme **mktemp** utilise les fonctions *mkstemp* et *mkdtemp* de la bibliothèque système C.

Compilez le paquet :

```
make
```

Installez le paquet :

```
make install
make install-tempfile
```

### 6.17.2. Contenu de Mktemp

**Programmes installés:** mktemp et tempfile

#### Descriptions courtes

**mktemp**      Crée des fichiers temporaires d'une façon sécurisée. Il est utilisé dans des scripts

**tempfile**    Crée des fichiers temporaires d'une façon moins sécurisée que **mktemp**. Il est installé pour des raisons de compatibilité descendante

## 6.18. Iana-Etc-1.04

Le paquet Iana-Etc fournit des données pour les services et protocoles réseau.

**Temps de construction estimé :** 0,1 SBU

**Espace disque requis :** 1,9 Mo

**Dépendances de l'installation :** Make

### 6.18.1. Installation et Iana-Etc

La commande suivante convertit les données brutes fournies par l'IANA dans les bons formats pour les fichiers de données `/etc/protocols` et `/etc/services` :

```
make
```

Installez le paquet :

```
make install
```

### 6.18.2. Contenu d'Iana-Etc

**Fichiers installés:** `/etc/protocols` et `/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

## 6.19. Findutils-4.2.23

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

**Temps de construction estimé :** 0,1 SBU

**Espace disque requis :** 9,4 Mo

**Dépendances de l'installation :** Bash, Binutils, Coreutils, Diffutils, GCC, Gettext, Glibc, Grep, Make et Sed

### 6.19.1. Installation de Findutils

Préparez la compilation de Findutils :

```
./configure --prefix=/usr --libexecdir=/usr/lib/locate \
--localstatedir=/var/lib/misc
```

Voici la signification de l'option de configure :

*--localstatedir*

Cette option modifie l'emplacement de la base de données **locate** avec `/var/lib/locate`, pour être compatible avec FHS.

Compilez le paquet :

```
make
```

Pour tester les résultats, lancez : **make check**.

Installez le paquet :

```
make install
```

### 6.19.2. Contenu de Findutils

**Programmes installés:** bigram, code, find, frcode, locate, updatedb et xargs

#### Descriptions courtes

<b>bigram</b>	Était auparavant utilisé pour créer les bases de données <b>locate</b>
<b>code</b>	Était auparavant utilisé pour créer les bases de données <b>locate</b> . Il est l'ancêtre de <b>frcode</b> .
<b>find</b>	Cherche dans les hiérarchies de répertoires donnés les fichiers correspondant à un critère spécifié
<b>frcode</b>	est appelé par <b>updatedb</b> pour compacter la liste des noms de fichiers. Il utilise front-compression, réduisant la taille de la base de données d'un facteur de 4 à 5
<b>locate</b>	recherche à travers la base de données des noms de fichiers et renvoie les noms contenant une certaine chaîne ou correspondant à un certain modèle
<b>updatedb</b>	met à jour la base de données <b>locate</b> . Il parcourt le système de fichiers entier (en incluant les

autres systèmes de fichiers actuellement montés, sauf si le contraire est spécifié) et place tous les noms de fichiers qu'ils trouvent dans la base de données

**xargs**

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



## 6.20. Gawk-3.1.4

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

**Temps de construction estimé :** 0,2 SBU

**Espace disque requis :** 16,4 Mo

**Dépendances de l'installation :** Bash, Binutils, Coreutils, Diffutils, GCC, Gettext, Glibc, Grep, Make et Sed

### 6.20.1. Installation de Gawk

Préparez la compilation de Gawk :

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

Compilez le paquet :

```
make
```

Pour tester les résultats, lancez : **make check**.

Installez le paquet :

```
make install
```

### 6.20.2. Contenu de Gawk

**Programmes installés:** awk (lien vers gawk), gawk, gawk-3.1.4, grcat, igawk, pgawk, pgawk-3.1.4 et pwcat

#### Descriptions courtes

<b>awk</b>	Un lien vers <b>gawk</b>
<b>gawk</b>	Un programme de manipulation de fichiers texte. C'est l'implémentation GNU d' <b>awk</b>
<b>gawk-3.1.4</b>	Un lien vers <b>gawk</b>
<b>grcat</b>	Sauvegarde la base de données des groupes, <code>/etc/group</code>
<b>igawk</b>	Donne à <b>gawk</b> la capacité d'inclure des fichiers
<b>pgawk</b>	La version de profilage de <b>gawk</b>
<b>pgawk-3.1.4</b>	Lien vers <b>pgawk</b>
<b>pwcat</b>	Sauvegarde la base de données des mots de passe, <code>/etc/passwd</code>

## 6.21. Ncurses-5.4

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

**Temps de construction estimé :** 0,6 SBU

**Espace disque requis :** 18,6 Mo

**Dépendances de l'installation :** Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Make et Sed

### 6.21.1. Installation de Ncurses

Préparez la compilation de Ncurses :

```
./configure --prefix=/usr --with-shared --without-debug
```

Compilez le paquet :

```
make
```

Ce paquet ne fournit pas de suite de tests.

Installez le paquet :

```
make install
```

Donnez les droits en exécution des bibliothèques Ncurses :

```
chmod -v 755 /usr/lib/*.5.4
```

Corrigez une bibliothèque qui ne devrait pas être exécutable :

```
chmod -v 644 /usr/lib/libncurses++.a
```

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

```
mv -v /usr/lib/libncurses.so.5* /lib
```

Comme les bibliothèques ont été déplacées, certains liens symboliques pointent vers des fichiers inexistantes. Re-créez ces liens symboliques :

```
ln -sfv ../../lib/libncurses.so.5 /usr/lib/libncurses.so  
ln -sfv libncurses.so /usr/lib/libcurses.so
```

## 6.21.2. Contenu de Ncurses

**Programmes installés:** captinfo (lien vers tic), clear, infocmp, infotocap (lien vers tic), reset (lien vers tset), tack, tic, toe, tput et tset

**Bibliothèques installées:** libcurses.[a,so] (lien vers libncurses.[a,so]), libform.[a,so], libmenu.[a,so], libncurses++.a, libncurses.[a,so] et libpanel.[a,so]

### Descriptions courtes

<b>captinfo</b>	Convertit une description termcap en description terminfo
<b>clear</b>	Efface l'écran si possible
<b>infocmp</b>	Compare ou affiche les descriptions terminfo
<b>infotocap</b>	Convertit une description terminfo en description termcap
<b>reset</b>	Réinitialise un terminal avec ses valeurs par défaut
<b>tack</b>	Vérificateur d'actions terminfo ; il est principalement utilisé pour tester la correction d'une entrée dans la base de données terminfo
<b>tic</b>	Le compilateur d'entrée de description terminfo, traduisant un fichier terminfo au format source dans un format binaire nécessaire pour les routines des bibliothèques ncurses. Un fichier terminfo contient des informations sur les capacités d'un terminal particulier
<b>toe</b>	Liste tous les types de terminaux disponibles, donnant pour chacun d'entre eux son nom principal et sa description
<b>tput</b>	Rend les valeurs de capacités dépendant du terminal disponibles au shell ; il peut aussi être utilisé pour réinitialiser un terminal ou pour afficher son nom long
<b>tset</b>	Peut être utilisé pour initialiser des terminaux
<code>libcurses</code>	Un lien vers <code>libncurses</code>
<code>libncurses</code>	Contient des fonctions pour afficher du texte de plusieurs façons compliquées sur un écran de terminal ; un bon exemple d'utilisation de ces fonctions est le menu affiché par le <b>make menuconfig</b> du noyau
<code>libform</code>	Contient des fonctions pour implémenter des formes
<code>libmenu</code>	Contient des fonctions pour implémenter des menus
<code>libpanel</code>	Contient des fonctions pour implémenter des panneaux

## 6.22. Readline-5.0

Le paquet Readline est un ensemble de bibliothèques qui offrent des fonctionnalités d'édition de la ligne de commande et d'historique.

**Temps de construction estimé :** 0,11 SBU

**Espace disque requis :** 9,1 Mo

**Dépendances de l'installation :** Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Make, Ncurses et Sed

### 6.22.1. Installation de Readline

Le correctif suivant inclut une correction pour un problème où Readline affiche parfois seulement 33 caractères sur une ligne puis continue sur la suivante. Il inclut aussi d'autres corrections recommandées par l'auteur de Readline.

```
patch -Np1 -i ../readline-5.0-fixes-1.patch
```

Préparez la compilation de Readline :

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

Compilez le paquet :

```
make SHLIB_XLDFLAGS=-lncurses
```

Voici la signification de l'option de make :

```
SHLIB_XLDFLAGS=-lncurses
```

Cette option force Readline à se lier à la bibliothèque `libncurses`.

Installez le paquet :

```
make install
```

Donnez aux bibliothèques dynamiques de Readline plus de droits appropriés :

```
chmod -v 755 /lib/lib{readline,history}.so*
```

Maintenant, déplacez les bibliothèques dynamiques à un emplacement plus appropriées :

```
mv -v /lib/lib{readline,history}.a /usr/lib
```

Ensuite, supprimez les fichiers `.so` dans `/lib` et créez un lien vers `/usr/lib`.

```
rm -v /lib/lib{readline,history}.so
ln -sfv ../../lib/libreadline.so.5 /usr/lib/libreadline.so
ln -sfv ../../lib/libhistory.so.5 /usr/lib/libhistory.so
```

## 6.22.2. Contenu de Readline

**Bibliothèques installées:** libhistory.[a,so] et libreadline.[a,so]

### Descriptions courtes

`libhistory` Fournit une interface utilisateur cohérente pour rappeler des lignes dans l'historique

`libreadline` Aide à une cohérence dans l'interface utilisateur pour des programmes discrets qui ont besoin d'une interface en ligne de commande

## 6.23. Vim-6.3

Le paquet Vim contient un puissant éditeur de texte.

**Temps de construction estimé :** 0,4 SBU

**Espace disque requis :** 38 Mo

**Dépendances de l'installation :** Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Grep, Make, Ncurses et Sed



### Alternatives à Vim

Si vous préférez un autre éditeur, comme Emacs, Joe ou Nano, merci de vous référer à <http://www.linuxfromscratch.org/blfs/view/svn/postlfs/editors.html> pour des instructions d'installation.

### 6.23.1. Installation de Vim

Tout d'abord, déballez les archives `vim-6.3.tar.bz2` et (en option) `vim-6.3-lang.tar.gz` dans le même répertoire. Puis, changez l'emplacement par défaut du fichier de configuration `vimrc` /etc :

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

Vim a deux vulnérabilités de sécurité déjà adressées par le mainteneur. Le correctif suivant s'occupe de ces problèmes :

```
patch -Np1 -i ../vim-6.3-security_fix-2.patch
```

Maintenant, préparez la compilation de Vim :

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

Voici la signification de l'option de configure :

*--enable-multibyte*

Ce commutateur optionnel mais hautement recommandé inclut le support pour l'édition de fichiers comprenant des codages de caractères multi-octets. 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 Core qui utilise UTF-8 comme ensemble de caractères par défaut.

Compilez le paquet :

```
make
```

Pour tester les résultats, lancez : `make test`. Néanmoins, cette suite de tests affiche à l'écran beaucoup de caractères binaires qui peuvent causer des soucis sur votre terminal. Ceci peut se résoudre en redirigeant la sortie vers un journal de traces.

Installez le paquet :

```
make install
```

Beaucoup d'utilisateurs sont habitués à utiliser `vi` au lieu de `vim`. Pour permettre l'exécution de `vim` quand

les utilisateurs saisissent habituellement **vi**, créez un lien symbolique :

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

Si un système X Window va être installé sur votre système LFS, il pourrait être nécessaire de recompiler Vim après avoir installé X. Vim fournit alors une jolie version GUI de l'éditeur qui requiert X et quelques autres bibliothèques pour s'installer. Pour plus d'informations sur ce processus, référez-vous à la documentation de Vim et à la page d'installation de Vim dans le livre BLFS sur <http://www.linuxfromscratch.org/blfs/view/svn/postlfs/editors.html#postlfs-editors-vim>.

## 6.23.2. 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 /etc/vimrc

set nocompatible
set backspace=2
syntax on
if (&term == "iterm") || (&term == "putty")
    set background=dark
endif

" Fin /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'
```

## 6.23.3. Contenu de Vim

**Programmes installés:** *efm\_filter.pl*, *efm\_perl.pl*, *ex* (lien vers vim), *less.sh*, *mve.awk*, *pltags.pl*, *ref*, *rview* (lien vers vim), *rvim* (lien vers vim), *shtags.pl*, *tcltags*, *vi* (lien vers vim), *view* (lien vers vim), *vim*, *vim132*, *vim2html.pl*, *vimdiff* (lien vers vim), *vimm*, *vimspell.sh*, *vimtutor* et *xxd*

### Descriptions courtes

<b>efm_filter.pl</b>	Un filtre pour créer un fichier d'erreur pouvant être lu par <b>vim</b>
<b>efm_perl.pl</b>	Reformate les messages d'erreur de l'interpréteur Perl pour utiliser le mode « quickfix » de <b>vim</b>

<b>ex</b>	Lance <b>vim</b> en mode ex
<b>less.sh</b>	Un script qui exécute <b>vim</b> avec less.vim
<b>mve.awk</b>	Traite les erreurs de <b>vim</b>
<b>pltags.pl</b>	Crée un fichier de balises pour le code Perl, utilisé par <b>vim</b>
<b>ref</b>	Vérifie la validité des arguments
<b>rview</b>	Une version restreinte de <b>view</b> : aucune commande shell ne peut être lancée et <b>view</b> ne peut pas être suspendu
<b>rvim</b>	Une version restreinte de <b>vim</b> : aucune commande shell ne peut être lancée et <b>vim</b> ne peut pas être suspendu
<b>shtags.pl</b>	Génère un fichier de balises pour les scripts Perl
<b>tcltags</b>	Génère un fichier de balises pour le code TCL
<b>view</b>	Lance <b>vim</b> en mode lecture seule
<b>vi</b>	L'éditeur
<b>vim</b>	L'éditeur
<b>vim132</b>	Lance <b>vim</b> avec le terminal en mode 132 colonnes
<b>vim2html.pl</b>	Convertit la documentation de Vim en HTML
<b>vimdiff</b>	Édite deux ou trois versions d'un fichier avec <b>vim</b> et montre les différences
<b>vimm</b>	Active le modèle d'entrée DEC sur un terminal distant
<b>vimspell.sh</b>	Un script qui vérifie un fichier et génère les instructions de syntaxe nécessaires pour mettre en surlignage dans <b>vim</b> . Ce script requiert l'ancienne commande Unix <b>spell</b> , qui n'est fourni ni par LFS ni par BLFS
<b>vimtutor</b>	Vous apprend les touches et les commandes basiques de <b>vim</b>
<b>xxd</b>	Fait un affichage hexa du fichier donné. Il peut aussi faire l'inverse pour une correspondance binaire



## 6.24. M4-1.4.3

Le paquet M4 contient un processeur de macros.

**Temps de construction estimé :** 0,1 SBU

**Espace disque requis :** 2,8 Mo

**Dépendances de l'installation :** Bash, Binutils, Coreutils, Diffutils, GCC, Gettext, Glibc, Grep, Make, Perl et Sed

### 6.24.1. Installation de M4

Préparez la compilation de M4 :

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

Compilez le paquet :

```
make
```

Pour tester les résultats, lancez : **make check**.

Installez le paquet :

```
make install
```

### 6.24.2. 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.

## 6.25. Bison-2.0

Le paquet Bison contient un générateur d'analyseurs.

**Temps de construction estimé :** 0,6 SBU

**Espace disque requis :** 9,9 Mo

**Dépendances de l'installation :** Bash, Binutils, Coreutils, Diffutils, GCC, Gettext, Glibc, Grep, M4, Make et Sed

### 6.25.1. Installation de Bison

Préparez la compilation de Bison :

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

Compilez le paquet :

```
make
```

Pour tester les résultats, lancez : **make check**.

Installez le paquet :

```
make install
```

### 6.25.2. Contenu de Bison

**Programmes installés:** bison et yacc

**Bibliothèque installée:** liby.a

#### Descriptions courtes

- bison** Génère, à partir d'une série de règles, un programme d'analyse de structure de fichiers texte ; Bison est un remplacement pour Yacc (Yet Another Compiler Compiler)
- yacc** Un emballage pour **bison**, utile pour les programmes qui appellent toujours **yacc** à la place de **bison** ; il appelle **bison** avec l'option `-y`
- liby.a** La bibliothèque Yacc contenant des implémentations, compatible Yacc, des fonctions `yyerror` et `main` ; Cette bibliothèque n'est généralement pas très utile mais POSIX la réclame

## 6.26. Less-382

Le paquet Less contient un visualisateur de fichiers texte.

**Temps de construction estimé :** 0,1 SBU

**Espace disque requis :** 2,3 Mo

**Dépendances de l'installation :** Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Grep, Make, Ncurses et Sed

### 6.26.1. Installation de Less

Préparez la compilation de Less :

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

Installez le paquet :

```
make install
```

### 6.26.2. Contents of Less

**Installed programs:** less, lessecho et lesskey

#### Short Descriptions

<b>less</b>	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
<b>lessecho</b>	nécessaire pour étendre les meta-caractères, comme * et ?, dans les noms de fichiers de systèmes Unix
<b>lesskey</b>	utilisé pour spécifier les associations de touches pour <b>less</b>

## 6.27. Groff-1.19.1

Le paquet Groff contient des programmes de formatage de texte.

**Temps de construction estimé :** 0,5 SBU

**Espace disque requis :** 38,7 Mo

**Dépendances de l'installation :** Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Make et Sed

### 6.27.1. Installation de Groff

Groff s'attend à ce que la variable d'environnement `PAGE` contienne la taille par défaut du papier. Pour les habitants des États-Unis, `PAGE=letter` est approprié. Ailleurs, `PAGE=A4` est préférable.

Préparez la compilation de Groff :

```
PAGE=[taille_papier] ./configure --prefix=/usr
```

Compilez le paquet :

```
make
```

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
```

### 6.27.2. Contenu de Groff

**Programmes installés:** addftinfo, afmtodit, eqn, eqn2graph, geqn (lien vers eqn), grn, grodvi, groff, groffer, grog, grolbp, grolj4, grops, grotty, gtbl (lien vers tbl), hpftodit, indxbib, lkbib, lookbib, mmroff, neqn, nroff, pfbtops, pic, pic2graph, post-grohtml, pre-grohtml, refer, soelim, tbl, tfmtodit, troff et zsoelim (lien vers soelim)

#### Descriptions courtes

<b>addftinfo</b>	Lit un fichier de polices troff et ajoute quelques informations métriques supplémentaire sur la police qui est utilisé par le système <b>groff</b>
<b>afmtodit</b>	Crée un fichier de police à utiliser avec <b>groff</b> et <b>grops</b>
<b>geqn</b>	Compile les descriptions d'équations imbriquées dans les fichiers d'entrées de troff pour obtenir des commandes comprises par <b>troff</b>
<b>eqn2graph</b>	Convertit une équation EQN troff en une image améliorée
<b>eqn</b>	Un lien vers <b>eqn</b>
<b>grn</b>	Un préprocesseur <b>groff</b> pour les fichiers gremlin

<b>grodvi</b>	Un pilote pour <b>groff</b> qui produit un format dvi TeX
<b>groff</b>	Une interface au système de formatage de document groff. Normalement, il lance le programme <b>troff</b> et un post-processeur approprié au périphérique sélectionné
<b>groffer</b>	Affiche des fichiers groff et des pages man sur des terminaux X et tty
<b>grog</b>	Lit des fichiers et devine les options <i>-e</i> , <i>-man</i> , <i>-me</i> , <i>-mm</i> , <i>-ms</i> , <i>-p</i> , <i>-s</i> et <i>-t</i> de <b>groff</b> , requises pour l'impression des fichiers. Il indique la commande <b>groff</b> incluant ces options
<b>grolbp</b>	Un pilote <b>groff</b> pour les imprimantes Canon CAPSL (imprimantes laser de la série LBP-4 et LBP-8)
<b>grolj4</b>	Un pilote pour <b>groff</b> produisant une sortie au format PCL5, intéressant les imprimantes HP Laserjet 4
<b>grops</b>	Traduit la sortie de GNU <b>troff</b> en Postscript
<b>grotty</b>	Traduit la sortie de GNU <b>troff</b> en un format compatible pour les périphériques de type machine à écrire
<b>gtbl</b>	Un lien vers <b>tbl</b>
<b>hptodit</b>	Crée un fichier de polices à utiliser avec <b>groff -Tlj4</b> à partir d'un fichier métrique de police HP
<b>indxbib</b>	Crée un index inversé d'un fichier spécifié, index utilisé par les bases de données bibliographiques avec <b>refer</b> , <b>lookbib</b> et <b>lkbib</b>
<b>lkbib</b>	recherche dans les bases de données bibliographiques pour des références contenant certaines clés et indique toute référence trouvée
<b>lookbib</b>	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
<b>mmroff</b>	Un pré-processeur pour <b>groff</b>
<b>neqn</b>	Formate les équations pour une sortie ASCII ( <i>American Standard Code for Information Interchange</i> )
<b>nroff</b>	Un script qui émule la commande <b>nroff</b> en utilisant <b>groff</b>
<b>pfbtops</b>	Traduit une police Postscript au format <code>.pfb</code> en ASCII
<b>pic</b>	Compile les descriptions d'images embarquées à l'intérieur de fichiers d'entrées troff ou TeX en des commandes comprises par TeX ou <b>troff</b>
<b>pic2graph</b>	Convertit un diagramme PIC en une image améliorée
<b>post-grohtml</b>	Traduit la sortie de GNU troff <b>troff</b> en html
<b>pre-grohtml</b>	Traduit la sortie de GNU <b>troff</b> en html
<b>refer</b>	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
<b>soelim</b>	Lit des fichiers et remplace les lignes de la forme <code>.so fichier</code> par le contenu du <i>fichier</i>

mentionné

<b>tbl</b>	Compile les descriptions des tables imbriquées dans les fichiers d'entrées troff en commandes comprises par <b>troff</b>
<b>tfmtoedit</b>	Crée un fichier de police à utiliser avec <b>groff -Tdvi</b>
<b>troff</b>	Est hautement compatible avec la commande Unix <b>troff</b> . Habituellement, il devrait être appelé en utilisant la commande <b>groff</b> , qui lance aussi les pré-processeurs et post-processeurs dans l'ordre approprié et avec les options appropriées
<b>zsoelim</b>	Un lien vers <b>soelim</b>

## 6.28. Sed-4.1.4

Le paquet Sed contient un éditeur de flux.

**Temps de construction estimé** : 0,2 SBU

**Espace disque requis** : 8,4 Mo

**Dépendances de l'installation** : Bash, Binutils, Coreutils, Diffutils, GCC, Gettext, Glibc, Grep, Make et Texinfo

### 6.28.1. Installation de Sed

Par défaut, Sed installe sa documentation HTML dans `/usr/share/doc`. Modifiez ceci par `/usr/share/doc/sed-4.1.4` en exécutant la commande **sed** suivante :

```
sed -i 's@/doc@&/sed-4.1.4@' doc/Makefile.in
```

Préparez la compilation de Sed :

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

Compilez le paquet :

```
make
```

Pour tester les résultats, lancez : **make check**.

Installez le paquet:

```
make install
```

### 6.28.2. Contenu de Sed

**Programme installé**: sed

#### Descriptions courtes

**sed** Filtre et transforme des fichiers texte en une seule passe

## 6.29. Flex-2.5.31

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

**Temps de construction estimé :** 0,1 SBU

**Espace disque requis :** 22,5 Mo

**Dépendances de l'installation :** Bash, Binutils, Bison, Coreutils, Diffutils, GCC, Gettext, Glibc, Grep, M4, Make et Sed

### 6.29.1. Installation de Flex

Flex contient quelques bogues connus. Corrigez-les avec le correctif suivant :

```
patch -Np1 -i ../flex-2.5.31-debian_fixes-3.patch
```

GNU autotools détecte que le code source de Flex a été modifié par le correctif précédent et essaie de mettre à jour la page man en accord. Ceci ne fonctionne pas sur beaucoup de systèmes et la page par défaut est bonne, donc assurez-vous qu'elle ne soit pas régénérée :

```
touch doc/flex.1
```

Préparez la compilation de Flex :

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

Compilez le paquet :

```
make
```

Pour tester les résultats, lancez : **make check**.

Installez le paquet :

```
make install
```

Quelques paquets s'attendent à trouver la bibliothèque `lex` dans `/usr/lib`. Créez un lien symbolique pour en tenir compte :

```
ln -sv libfl.a /usr/lib/libl.a
```

Quelques programmes ne connaissent pas encore **flex** et essaient de lancer son prédécesseur **lex**. Pour supporter ces programmes, créez un script d'emballage nommé `lex` appelant `flex` en mode d'émulation **lex** :

```
cat > /usr/bin/lex << "EOF"
#!/bin/sh
# Begin /usr/bin/lex

exec /usr/bin/flex -l "$@"

# End /usr/bin/lex
EOF
chmod -v 755 /usr/bin/lex
```

### 6.29.2. Contenu de Flex



**Programmes installés:** flex et lex

**Bibliothèque installée:** libfl.a

## Descriptions courtes

**flex** Un outil pour générer des programmes reconnaissant des modèles dans un texte ; cela permet une grande diversité pour spécifier les règles de recherche de modèle, éradiquant ainsi le besoin de développer un programme spécialisé

**lex** Un script qui exécute **flex** en mode d'émulation **lex**

`libfl.a` La bibliothèque `flex`

## 6.30. Gettext-0.14.3

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

**Temps de construction estimé :** 1,2 SBU

**Espace disque requis :** 65,1 Mo

**Dépendances de l'installation :** Bash, Binutils, Bison, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Make et Sed

### 6.30.1. Installation de Gettext

Préparez la compilation de Gettext :

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

Compilez le paquet :

```
make
```

Pour tester les résultats, lancez : **make check**. Ceci peut prendre beaucoup de temps, pratiquement 7 SBU.

Installez le paquet :

```
make install
```

### 6.30.2. Contenu de Gettext

**Programmes installés:** autopoint, config.charset, config.rpath, envsubst, gettext, gettextize, hostname, msgattrib, msgcat, msgcmp, msgcomm, msgconv, msgen, msgexec, msgfilter, msgfmt, msggrep, msginit, msgmerge, msgunfmt, msguniq, ngettext et xgettext

**Bibliothèques installées:** libasprintf.[a,so], libgettextlib.so, libgettextpo.[a,so] et libgettextsrc.so

#### Descriptions courtes

<b>autopoint</b>	Copie les fichiers d'infrastructure standard gettext en un paquet source
<b>config.charset</b>	Affiche une table des caractères dépendante du système
<b>config.rpath</b>	Affiche un ensemble de variables dépendant du système, décrivant comment initialiser le chemin de recherche à l'exécution des bibliothèques partagées dans un exécutable
<b>envsubst</b>	Substitue les variables d'environnement dans des chaînes de format shell
<b>gettext</b>	Traduit un message en langue naturelle dans la langue de l'utilisateur en recherchant la traduction dans un catalogue de messages
<b>gettextize</b>	Copie tous les fichiers standard Gettext dans le répertoire de haut niveau d'un paquet, pour commencer son internationalisation
<b>hostname</b>	Affiche un nom d'hôte réseau sous plusieurs formats

<b>msgattrib</b>	Filtre les messages d'un catalogue de traduction suivant leurs attributs et manipule les attributs
<b>msgcat</b>	Concatène et fusionne les fichiers <code>.po</code> donnés
<b>msgcmp</b>	Compare deux fichiers <code>.po</code> pour vérifier que les deux contiennent le même ensemble de chaînes msgid
<b>msgcomm</b>	trouve les messages qui sont communs aux fichiers <code>.po</code>
<b>msgconv</b>	Convertit un catalogue de traduction en un autre codage de caractères
<b>msgen</b>	Crée un catalogue de traduction anglais
<b>msgexec</b>	Applique une commande pour toutes les traductions d'un catalogue de traduction
<b>msgfilter</b>	Applique un filtre à toutes les traductions d'un catalogue de traductions
<b>msgfmt</b>	Génère un catalogue binaire de messages à partir d'un catalogue de traductions
<b>msggrep</b>	Extrait tous les messages d'un catalogue de traductions correspondant à un modèle donné ou appartenant à d'autres sources données
<b>msginit</b>	Crée un nouveau fichier <code>.po</code> , initialisant la méta-information avec des valeurs de l'environnement de l'utilisateur
<b>msgmerge</b>	Combine deux traductions brutes en un seul fichier
<b>msgunfmt</b>	Décompile un catalogue de messages binaire en un texte brut de la traduction
<b>msguniq</b>	Unifie les traductions dupliquées en un catalogue de traduction
<b>ngettext</b>	Affiche les traductions dans la langue native d'un message texte dont la forme grammaticale dépend d'un nombre
<b>xgettext</b>	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 <code>&lt;string&gt;</code> et les flux <code>&lt;iostream&gt;</code>
<code>libgettextlib</code>	Une bibliothèque privée contenant les routines communes utilisées par les nombreux programmes gettext. Ils ne sont pas fait pour une utilisation générale
<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 livrés avec Gettext ne vont pas suffire (comme <b>msgcomm</b> , <b>msgcmp</b> , <b>msgattrib</b> et <b>msgen</b> )
<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

## 6.31. Inetutils-1.4.2

Le paquet Inetutils contient des programmes réseau basiques.

**Temps de construction estimé :** 0,2 SBU

**Espace disque requis :** 8,7 Mo

**Dépendances de l'installation :** Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Grep, Make, Ncurses et Sed

### 6.31.1. Installation de Inetutils

Inetutils a quelques soucis avec la série 2.6 du noyau Linux. Corrigez-les en appliquant le correctif suivant :

```
patch -Np1 -i ../inetutils-1.4.2-kernel_headers-1.patch
```

Les programmes venant avec Inetutils ne seront pas tous installés. Néanmoins, le système de construction d'Inetutils insistera malgré tout sur l'installation de toutes les pages man. Le correctif suivant corrigera cette situation :

```
patch -Np1 -i ../inetutils-1.4.2-no_server_man_pages-1.patch
```

Préparez la compilation d'Inetutils :

```
./configure --prefix=/usr --libexecdir=/usr/sbin \
  --sysconfdir=/etc --localstatedir=/var \
  --disable-logger --disable-syslogd \
  --disable-whois --disable-servers
```

Voici la signification des options de configure :

*--disable-logger*

Cette option empêche l'installation du programme **logger** par Inetutils. Ce programme est utilisé par les scripts pour passer des messages au démon des traces système. Nous ne l'installons pas car Util-linux livre une version bien meilleure après.

*--disable-syslogd*

Cette option empêche l'installation du démon de traces système par Inetutils car il est installé avec le paquet Sysklogd.

*--disable-whois*

cette option désactive la construction du client **whois** d'Inetutils qui est vraiment obsolète. Les instructions pour un meilleur client **whois** sont dans le livre BLFS.

*--disable-servers*

Ceci désactive l'installation des différents serveurs réseau inclus dans le paquet Inetutils. Ces serveurs semblent inappropriés dans un système LFS de base. Certains sont non sécurisés et ne sont pas considérés sains sur des réseaux de confiance. Plus d'informations sont disponibles sur <http://www.linuxfromscratch.org/blfs/view/svn/basicnet/inetutils.html>. Notez que de meilleurs remplacements sont disponibles pour certains de ces serveurs.

Compilez le paquet :

```
make
```

Installez le paquet :

```
make install
```

Déplacez le programme **ping** à un emplacement compatible avec FHS :

```
mv -v /usr/bin/ping /bin
```

## 6.31.2. Contenu d'Inetutils

**Programmes installés:** ftp, ping, rcp, rlogin, rsh, talk, telnet et tftp

### Descriptions courtes

<b>ftp</b>	Est un programme pour le protocole de transfert de fichier
<b>ping</b>	Envoie des paquets echo-request et affiche le temps mis pour que la réponse arrive
<b>rcp</b>	Fait une copie de fichiers distants
<b>rlogin</b>	Permet une connexion à distance
<b>rsh</b>	Exécute un shell distant
<b>talk</b>	Est utilisé pour discuter avec un autre utilisateur
<b>telnet</b>	Une interface du protocole TELNET
<b>tftp</b>	Un programme de transfert trivial de fichiers

## 6.32. IPRoute2-2.6.11-050330

Le paquet IPRoute2 contient des programmes pour le réseau, basique ou avancé, basé sur IPV4.

**Temps de construction estimé :** 0,1 SBU

**Espace disque requis :** 4,3 Mo

**Dépendances de l'installation :** GCC, Glibc, Make, Linux-Headers et Sed

### 6.32.1. Installation d'IPRoute2

Le binaire **arpd** inclus dans ce paquet est dépendant de Berkeley DB. Comme **arpd** n'est pas un prérequis commun sur un système Linux de base, supprimez la dépendance sur Berkeley DB en exécutant la commande **sed** ci-dessous. Si le binaire **arpd** est nécessaire, les instructions pour compiler Berkeley DB sont disponibles dans le livre BLFS sur <http://www.linuxfromscratch.org/blfs/view/svn/server/databases.html#db>.

```
sed -i '/^TARGETS/s@arpd@g' misc/Makefile
```

Préparez la compilation d'IPRoute2 :

```
./configure
```

Compilez le paquet :

```
make SBINDIR=/sbin
```

Voici la signification de l'option de make :

```
SBINDIR=/sbin
```

Ceci nous assure que les binaires IPRoute2 seront installés dans `/sbin`. C'est le bon emplacement suivant la FHS parce que certains des binaires IPRoute2 sont utilisés dans les scripts de démarrage.

Installez le paquet :

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

### 6.32.2. Contenu d'IPRoute2

**Programmes installés:** ctstat (lien vers lstat), ifcfg, ifstat, ip, lstat, nstat, routef, routel, rtacct, rtmon, rtpr, rtstat (lien vers lstat), ss et tc.

#### Descriptions courtes

<b>ctstat</b>	Outil donnant le statut de la connexion
<b>ifcfg</b>	Un emballage en script shell pour la commande <b>ip</b>
<b>ifstat</b>	Affiche les statistiques des interfaces, incluant le nombre de de paquets émis et transmis par l'interface.
<b>ip</b>	L'exécutable principal. Il a plusieurs fonctions : <b>ip link [périphérique]</b> autorise les utilisateurs à regarder l'état des périphériques et à faire des changements.

**ip addr** autorise les utilisateurs à regarder les adresses et leurs propriétés, à ajouter de nouvelles adresses et à supprimer les anciennes.

**ip neighbor** autorise les utilisateurs à regarder dans les liens des voisins et dans leurs propriétés, à ajouter de nouvelles entrées et à supprimer les anciennes.

**ip rule** autorise les utilisateurs à regarder les politiques de routage et à les modifier.

**ip route** autorise les utilisateurs à regarder la table de routage et à modifier les règles de routage.

**ip tunnel** autorise les utilisateurs à regarder les tunnels IP et leurs propriétés, et à les modifier.

**ip maddr** autorise les utilisateurs à regarder adresses multicast et leurs propriétés, et à les changer.

**ip mroute** autorise les utilisateurs à configurer, modifier ou supprimer le routage multicast.

**ip monitor** autorise les utilisateurs à surveiller en continu l'état des périphériques, des adresses et des routes.

<b>lnstat</b>	Fournit les statistiques réseau Linux. C'est un remplacement plus généraliste et plus complet de l'ancien programme <b>rtstat</b>
<b>nstat</b>	Affiche les statistiques réseau.
<b>route</b>	Un composant de <b>ip route</b> pour vider les tables de routage.
<b>route</b>	Un composant de <b>ip route</b> pour afficher les tables de routage.
<b>rtacct</b>	Affiche le contenu de <code>/proc/net/rt_acct</code>
<b>rtmon</b>	Outil de surveillance de routes.
<b>rtpr</b>	Convertit la sortie de <b>ip -o</b> en un format lisibles
<b>rtstat</b>	Outil de statut de routes
<b>ss</b>	Similaire à la commande <b>netstat</b> ; affiche les connexions actives
<b>tc</b>	Exécutable de contrôle du trafic ; utile pour l'implémentation de la qualité de service (QOS) et de la classe de service (COS)
	<b>tc qdisc</b> autorise les utilisateurs à configurer la discipline de queues
	<b>tc class</b> autorise les utilisateurs à configurer les classes suivant la planification de la discipline de queues
	<b>tc estimator</b> autorise les utilisateurs à estimer le flux réseau dans un réseau
	<b>tc filter</b> autorise les utilisateurs à configurer les filtres de paquets pour QOS/COS
	<b>tc policy</b> autorise les utilisateurs à configurer les politiques QOS/COS

## 6.33. Perl-5.8.7

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

**Temps de construction estimé :** 4,1 SBU

**Espace disque requis :** 140 Mo

**Dépendances de l'installation :** Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Make et Sed

### 6.33.1. Installation de Perl

Si vous voulez avoir un contrôle total sur la façon dont Perl est configuré, lancez le script interactif **Configure** et choisissez la façon dont le paquet est construit. Si les valeurs par défaut détectées automatiquement sont convenables, préparez la compilation de Perl ainsi :

```
./configure.gnu --prefix=/usr -Dpager="/bin/less -isR"
```

Voici la signification de l'option de configure :

```
-Dpager="/bin/less -isR"
```

Ceci corrige une erreur dans la façon dont **perldoc** fait appel au programme **less**.

Compilez le paquet :

```
make
```

Pour exécuter la suite de tests, créez tout d'abord un fichier `/etc/hosts` basique, nécessaire à quelques tests pour résoudre le nom *localhost* :

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

Maintenant, lancez les tests si vous le souhaitez :

```
make test
```

Installez le paquet :

```
make install
```

### 6.33.2. Contenu de Perl

**Programmes installés:** a2p, c2ph, dprofpp, enc2xs, find2perl, h2ph, h2xs, libnetcfg, perl, perl5.8.7 (lien vers perl), perlbug, perlcc, perldoc, perlivp, piconv, pl2pm, pod2html, pod2latex, pod2man, pod2text, pod2usage, podchecker, podselect, psed (lien vers s2p), pstruct (lien vers c2ph), s2p, splain et xsubpp

**Bibliothèques installées:** Quelques centaines qui ne pourraient pas toutes être listées ici

#### Descriptions courtes

**a2p** Traduit awk en perl

**c2ph** Affiche les structures C comme si elles étaient générées à partir de **cc -g -S**

**dprofpp** Affiche les données profile de Perl



<b>en2cxs</b>	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>
<b>find2perl</b>	Traduit les commandes <b>find</b> en Perl
<b>h2ph</b>	Convertit les fichiers d'en-têtes C <code>.h</code> en fichiers d'en-têtes Perl <code>.ph</code>
<b>h2xs</b>	Convertit les fichiers d'en-têtes C <code>.h</code> en extensions Perl
<b>libnetcfg</b>	Peut être utilisé pour configurer <code>libnet</code>
<b>perl</b>	Combine quelques-unes des meilleures fonctionnalités de C, <b>sed</b> , <b>awk</b> et <b>sh</b> en un langage style couteau suisse
<b>perl5.8.7</b>	Un lien vers <b>perl</b>
<b>perlbug</b>	Utilisé pour générer des rapports de bogues sur Perl ou les modules l'accompagnant et pour les envoyer par courrier électronique
<b>perlcc</b>	Génère des exécutables à partir des programmes Perl
<b>perldoc</b>	Affiche une partie de la documentation au format pod, embarquée dans le répertoire d'installation de Perl ou dans un script Perl
<b>perlivp</b>	La procédure de vérification d'installation de Perl ( <i>Perl Installation Verification Procedure</i> ). Il peut être utilisé pour vérifier que Perl et ses bibliothèques ont été installés correctement
<b>piconv</b>	une version Perl du convertisseur de codage des caractères <b>iconv</b>
<b>pl2pm</b>	un outil simple pour la conversion des fichiers Perl4 <code>.pl</code> en modules Perl5 <code>.pm</code>
<b>pod2html</b>	Convertit des fichiers à partir du format pod vers le format HTML
<b>pod2latex</b>	Convertit des fichiers à partir du format pod vers le format LaTeX
<b>pod2man</b>	Convertit des fichiers à partir du format pod vers une entrée formatée <code>*roff</code>
<b>pod2text</b>	Convertit des fichiers à partir du format pod vers du texte ANSI
<b>pod2usage</b>	Affiche les messages d'usage à partir des documents embarqués pod
<b>podchecker</b>	Vérifie la syntaxe du format pod des fichiers de documentation
<b>podselect</b>	Affiche les sections sélectionnées de la documentation pod
<b>psed</b>	Une version Perl de l'éditeur en flux <b>sed</b>
<b>pstruct</b>	affiche les structures C générées à partir de <b>cc -g -S</b>
<b>s2p</b>	Traduit les scripts <b>sed</b> en perl
<b>splain</b>	est utilisé pour forcer la verbosité des messages d'avertissement avec Perl
<b>xsubpp</b>	Convertit le code Perl XS en code C

## 6.34. Texinfo-4.8

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

**Temps de construction estimé :** 0,2 SBU

**Espace disque requis :** 14,7 Mo

**Dépendances de l'installation :** Bash, Binutils, Coreutils, Diffutils, GCC, Gettext, Glibc, Grep, Make, Ncurses et Sed

### 6.34.1. Installation de Texinfo

Texinfo permet aux utilisateurs locaux d'écraser des fichiers arbitraires avec une attque de lien symbolique sur les fichiers temporaires. Appliquez le correctif suivant pour supprimer ce problème :

```
patch -Np1 -i ../texinfo-4.8-tempfile_fix-1.patch
```

Préparez la compilation de Texinfo :

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

Compilez le paquet :

```
make
```

Pour tester les résultats, lancez : **make check**.

Installez le paquet :

```
make install
```

De manière optionnelle, installez les composants appartenant à une installation TeX :

```
make TEXMF=/usr/share/texmf install-tex
```

Voici la signification du paramètre de make :

```
TEXMF=/usr/share/texmf
```

La variable TEXMF du Makefile contient l'emplacement de la racine de votre répertoire TeX si, par exemple, un paquet TeX sera installé plus tard.

Le système de documentation Info utilise un fichier texte pour contenir sa liste des entrées de menu. Le fichier est situé dans `/usr/share/info/dir`. Malheureusement, à cause de problèmes occasionnels dans les Makefiles de différents paquets, il peut être non synchronisé avec les pages info. Si le fichier `/usr/share/info/dir` a besoin d'être re-créé, les commandes suivantes accompliront cette tâche :

```
cd /usr/share/info
rm dir
for f in *
do install-info $f dir 2>/dev/null
done
```

## 6.34.2. Contenu de Texinfo

**Programmes installés:** info, infokey, install-info, makeinfo, texi2dvi, texi2pdf et texindex

### Descriptions courtes

<b>info</b>	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 <b>man bison</b> et <b>info bison</b> .
<b>infokey</b>	Compile un fichier source contenant des personnalisations Info en un format binaire
<b>install-info</b>	Utilisé pour installer les pages info ; il met à jour les entrées dans le fichier index d' <b>info</b>
<b>makeinfo</b>	Traduit les sources Texinfo données dans différents autres langages : pages info, texte ou HTML
<b>texi2dvi</b>	Utilisé pour formater le document Texinfo indiqué en un fichier indépendant des périphériques, pouvant être édité
<b>texi2pdf</b>	Utilisé pour formater le document Texinfo indiqué en un fichier PDF
<b>texindex</b>	Utilisé pour trier les fichiers d'index de Texinfo

## 6.35. Autoconf-2.59

Le paquet Autoconf contient des programmes produisant des scripts shell qui configurent automatiquement le code source.

**Temps de construction estimé :** 0,5 SBU

**Espace disque requis :** 8,5 Mo

**Dépendances de l'installation :** Bash, Coreutils, Diffutils, Grep, M4, Make, Perl et Sed

### 6.35.1. Installation d'Autoconf

Préparez la compilation d'Autoconf :

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

Compilez le paquet :

```
make
```

Pour tester les résultats, lancez : **make check**. Ceci prend du temps, pratiquement 2 SBU.

Installez le paquet :

```
make install
```

### 6.35.2. Contenu de Autoconf

**Programmes installés:** autoconf, autoheader, autom4te, autoreconf, autoscan, autoupdate et ifnames

#### Descriptions courtes

<b>autoconf</b>	Produit des scripts shell configurant automatiquement des paquets de code source, permettant ainsi de les adapter à tous les types de systèmes Unix. Les scripts de configuration qu'il produit sont indépendants. Les exécuter ne nécessite pas le programme <b>autoconf</b> .
<b>autoheader</b>	Un outil pour créer des fichiers modèle d'instructions <i>C #define</i> que configure utilise.
<b>autom4te</b>	Un emballage pour le processeur de macro M4.
<b>autoreconf</b>	Exécute automatiquement <b>autoconf</b> , <b>autoheader</b> , <b>aclocal</b> , <b>automake</b> , <b>gettextize</b> et <b>libtoolize</b> dans le bon ordre pour gagner du temps lorsque des modifications ont eu lieu sur les fichiers modèles d' <b>autoconf</b> et d' <b>automake</b> .
<b>autoscan</b>	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> utilisant <code>configure.in</code> comme base pour le paquet
<b>autoupdate</b>	Modifie un fichier <code>configure.in</code> qui appelle toujours les macros <b>autoconf</b> par leurs anciens noms pour qu'il utilise les noms de macros actuels.

**ifnames**

Sert à écrire les fichiers `configure.in` pour un paquet logiciel. Il affiche les identifiants que le paquet utilise dans des conditions du préprocesseur C. Si un paquet a déjà été initialisé pour avoir une certaine portabilité, ce programme aide à déterminer ce que **configure** doit vérifier. Il peut aussi remplir les blancs dans un fichier `configure.in` généré par **autoscan**.

## 6.36. Automake-1.9.5

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

**Temps de construction estimé :** 0,2 SBU

**Espace disque requis :** 8,8 Mo

**Dépendances de l'installation :** Autoconf, Bash, Coreutils, Diffutils, Grep, M4, Make, Perl et Sed

### 6.36.1. Installation de Automake

Préparez la compilation d'Automake :

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

Compilez le paquet :

```
make
```

Pour tester les résultats, lancez : **make check**. Ceci peut prendre beaucoup de temps, environ 5 SBU.

Installez le paquet :

```
make install
```

### 6.36.2. Contenu d'Automake

**Programmes installés:** acinstall, aclocal, aclocal-1.9.5, automake, automake-1.9.5, compile, config.guess, config.sub, depcomp, elisp-comp, install-sh, mdate-sh, missing, mkinstalldirs, py-compile, symlink-tree et ylwrap

#### Descriptions courtes

<b>acinstall</b>	Un script qui installe des fichiers M4, style aclocal
<b>aclocal</b>	Génère des fichiers <code>aclocal.m4</code> basés sur le contenu de fichiers <code>configure.in</code>
<b>aclocal-1.9.5</b>	Un lien vers <b>aclocal</b>
<b>automake</b>	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> correspondant
<b>automake-1.9.5</b>	Un lien vers <b>automake</b>
<b>compile</b>	Un emballage pour les compilateurs
<b>config.guess</b>	Un script qui tente de deviner un triplet canonique pour la construction donnée, l'hôte ou l'architecture de la cible
<b>config.sub</b>	Un script contenant une sous-routine de validation de configuration

<b>depcomp</b>	Un script pour compiler un programme de façon à ce que les informations de dépendances soient générées en plus de la sortie désirée
<b>elisp-compile</b>	Compile le code Lisp d'Emacs
<b>install-sh</b>	Un script qui installe un programme, un script ou un fichier de données
<b>mdate-sh</b>	Un script qui affiche la date de modification d'un fichier ou répertoire
<b>missing</b>	Un script agissant comme remplaçant pour les programmes GNU manquants lors d'une installation
<b>mkinstalldirs</b>	Un script qui crée un ensemble de répertoires
<b>py-compile</b>	Compile un programme Python
<b>symlink-tree</b>	Un script créant un ensemble de liens à partir d'un ensemble de répertoires
<b>ylwrap</b>	Un emballage pour <b>lex</b> et <b>yacc</b>

## 6.37. Bash-3.0

Le paquet Bash contient le shell Bourne-Again.

**Temps de construction estimé :** 1,2 SBU

**Espace disque requis :** 20,6 Mo

**Dépendances de l'installation :** Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Make, Ncurses et Sed.

### 6.37.1. Installation de Bash

Si vous avez téléchargé l'archive tar de la documentation de Bash et si vous souhaitez installer la documentation HTML, exécutez les commandes suivantes :

```
tar -xvf ../bash-doc-3.0.tar.gz &&
sed -i "s|htmldir = @htmldir@|htmldir = /usr/share/doc/bash-3.0|" \
    Makefile.in
```

Le correctif suivant corrige quelques problèmes, dont celui où Bash ne montre quelque fois que 33 caractères sur une ligne puis passe à la suivante :

```
patch -Np1 -i ../bash-3.0-fixes-3.patch
```

Bash a aussi des problèmes lorsqu'il est compilé avec les dernières versions de Glibc. Le correctif suivant résout ce problème :

```
patch -Np1 -i ../bash-3.0-avoid_WCONTINUED-1.patch
```

Préparez la compilation de Bash :

```
./configure --prefix=/usr --bindir=/bin \
    --without-bash-malloc --with-installed-readline
```

Voici la signification de l'option de configure :

*--with-installed-readline*

Ce commutateur indique à Bash d'utiliser la bibliothèque `readline` qui est déjà installée sur le système plutôt que d'utiliser sa propre version de `readline`.

Compilez le paquet :

```
make
```

Pour tester les résultats, lancez : `make tests`.

Installez le paquet :

```
make install
```

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

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



#### Note



Les paramètres utilisés font que **bash** lance un shell de connexion interactif et désactive le hachage, de façon à ce que les nouveaux programme soient découverts au fur et à mesure de leur disponibilité.

## 6.37.2. Contenu de Bash

**Programmes installés:** bash, bashbug et sh (lien vers bash)

### Descriptions courtes

- bash** Un interpréteur de commandes largement utilisé ; il réalise un grand nombre d'expansions et de substitutions sur une ligne de commande donnée avant de l'exécuter, rendant cet interpréteur très puissant
- bashbug** Un script shell pour aider l'utilisateur à composer et à envoyer des courriers électroniques contenant des rapports de bogues spécialement formatés concernant **bash**
- sh** Un lien symbolique vers le programme **bash** ; à son appel en tant que **sh**, **bash** essaie de copier le comportement initial des versions historiques de **sh** aussi fidèlement que possible, tout en se conformant aussi au standard POSIX

## 6.38. File-4.13

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

**Temps de construction estimé :** 0,1 SBU

**Espace disque requis :** 6,2 Mo

**Dépendances de l'installation :** Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Grep, Make, Sed et Zlib

### 6.38.1. Installation de File

Préparez la compilation de File :

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

Compilez le paquet :

```
make
```

Installez le paquet :

```
make install
```

### 6.38.2. Contenu de File

**Programmes installés:** file

**Bibliothèque installée:** libmagic.[a,so]

#### Descriptions courtes

- |                 |   |
|-----------------|---|
| <b>file</b>     | Tente de classifier chaque fichier donné. Il réalise ceci en exécutant différents tests : tests sur le système de fichiers, tests des nombres magiques et tests de langages |
| <b>libmagic</b> | Contient des routines pour la reconnaissance de nombres magiques utilisés par le programme <b>file</b>  |

## 6.39. Libtool-1.5.14

Le paquet Libtool contient le script de support de bibliothèques génériques GNU. Il emballe la complexité d'utilisation de bibliothèques partagées dans une interface cohérente et portable.

**Temps de construction estimé :** 1,5 SBU

**Espace disque requis :** 19,7 Mo

**Dépendances de l'installation :** Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Grep, Make et Sed

### 6.39.1. Installation de Libtool

Préparez la compilation de Libtool :

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

Compilez le paquet :

```
make
```

Pour tester les résultats, lancez : **make check**.

Installez le paquet :

```
make install
```

### 6.39.2. Contenu de Libtool

**Programmes installés:** libtool et libtoolize

**Bibliothèques installées:** libltdl.[a,so]

#### Descriptions courtes

<b>libtool</b>	Fournit des services de support de construction généralisée de bibliothèques
<b>libtoolize</b>	Fournit une façon standard d'ajouter le support de <b>libtool</b> dans un paquet
<b>libltdl</b>	Cache les nombreuses difficultés avec dlopen sur les bibliothèques

## 6.40. Bzip2-1.0.3

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

**Temps de construction estimé** : 0,1 SBU

**Espace disque requis** : 3,9 Mo

**Dépendances de l'installation** : Bash, Binutils, Coreutils, Diffutils, GCC, Glibc et Make

### 6.40.1. Installation de Bzip2

Appliquez un correctif pour installer la documentation de ce paquet :

```
patch -Np1 -i ../bzip2-1.0.3-install_docs-1.patch
```

La commande **bzgrep** n'échappe pas '|' et '&' dans les noms de fichiers qui lui sont passés. Ceci permet l'exécution de commandes arbitraires avec les droits de l'utilisateur exécutant **bzgrep**. Appliquez ce qui suit pour corriger cela :

```
patch -Np1 -i ../bzip2-1.0.3-bzgrep_security-1.patch
```

Préparez la compilation de Bzip2 avec :

```
make -f Makefile-libbz2_so
make clean
```

Le commutateur **-f** fera que Bzip2 sera construit en utilisant un fichier `Makefile` différent, dans ce cas le fichier `Makefile-libbz2_so`, qui crée une bibliothèque `libbz2.so` dynamique et lie les outils Bzip2 avec.

Compilez et testez le paquet :

```
make
```

En cas de réinstallation de Bzip2, effectuez un **rm -vf /usr/bin/bz\*** en premier, sinon les prochains **make install** échoueront.

Installez les programmes :

```
make install
```

Installez le binaire dynamique **bzip2** dans le répertoire `/bin`, créez les liens symboliques nécessaires et nettoyez :

```
cp -v bzip2-shared /bin/bzip2
cp -av libbz2.so* /lib
ln -sv ../../lib/libbz2.so.1.0 /usr/lib/libbz2.so
rm -v /usr/bin/{bunzip2,bzcat,bzip2}
ln -sv bzip2 /bin/bunzip2
ln -sv bzip2 /bin/bzcat
```

## 6.40.2. Contenu de Bzip2

**Programmes installés:** bunzip2 (lien vers bzip2), bzip2, bzip2recover, bzless et bzmores

**Bibliothèques installées:** libbz2.[a,so]

### Descriptions courtes

<b>bunzip2</b>	Décompresse les fichiers compressés avec bzip
<b>bzip2</b>	Décompresse vers la sortie standard
<b>bzcat</b>	Décompresse vers la sortie standard
<b>bzcmp</b>	Lance <b>cmp</b> sur des fichiers compressés avec bzip
<b>bzdiff</b>	Lance <b>diff</b> sur des fichiers compressés avec bzip
<b>bzgrep</b>	Lance <b>grep</b> sur des fichiers compressés avec bzip
<b>bzegrep</b>	Lance <b>egrep</b> sur des fichiers compressés avec bzip
<b>bzfgrep</b>	Lance <b>fgrep</b> sur des fichiers compressés avec bzip
<b>bzip2</b>	Comprime les fichiers en utilisant l'algorithme de compression de texte par tri de blocs de Burrows-Wheeler avec le codage Huffman ; le taux de compression est meilleur que celui auquel parviennent les outils de compression plus conventionnels utilisant les algorithmes « Lempel-Ziv », comme <b>gzip</b>
<b>bzip2recover</b>	Essaie de récupérer des données à partir de fichiers endommagés, compressés avec bzip
<b>bzless</b>	Lance <b>less</b> sur des fichiers compressés avec bzip
<b>bzmores</b>	Lance <b>mores</b> sur des fichiers compressés avec bzip
libbz2*	La bibliothèque implémentant la compression de données sans perte par tri de blocs, utilisant l'algorithme de Burrows-Wheeler

## 6.41. Diffutils-2.8.1

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

**Temps de construction estimé :** 0,1 SBU

**Espace disque requis :** 5,6 Mo

**Dépendances de l'installation :** Bash, Binutils, Coreutils, Diffutils, GCC, Gettext, Glibc, Grep, Make et Sed

### 6.41.1. Installation de Diffutils

Préparez la compilation de Diffutils :

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

Compilez le paquet :

```
make
```

Ce paquet ne fournit pas de suite de tests.

Installez ce paquet :

```
make install
```

### 6.41.2. Contenu de Diffutils

**Programmes installés:** cmp, diff, diff3 et sdiff

#### Descriptions courtes

- cmp** Compare deux fichiers et rapporte si ou à quels endroits ils diffèrent
- diff** Compare deux fichiers ou répertoires et rapporte les lignes où les fichiers diffèrent.
- diff3** Compare trois fichiers ligne par ligne
- sdiff** Assemble deux fichiers et affiche le résultat de façon interactive

## 6.42. Kbd-1.12

Le paquet Kbd contient les fichiers de plan de codage et des outils pour le clavier.

**Temps de construction estimé :** 0,1 SBU

**Espace disque requis :** 11,8 Mo

**Dépendances de l'installation :** Bash, Binutils, Bison, Coreutils, Diffutils, Flex, GCC, Gettext, Glibc, Grep, Gzip, M4, Make et Sed

### 6.42.1. Installation de Kbd

Préparez la compilation de Kbd :

```
./configure
```

Compilez le paquet :

```
make
```

Maintenant, installez-le :

```
make install
```

### 6.42.2. Contenu de Kbd

**Programmes installés:** chvt, dealloctv, dumpkeys, fgconsole, getkeycodes, getunimap, kbd\_mode, kbdrate, loadkeys, loadunimap, mapscrn, openvt, psfaddtable (lien vers psfxtable), psfgettable (lien vers psfxtable), psfstriptable (lien vers psfxtable), psfxtable, resizecons, setfont, setkeycodes, setleds, setlogcons, setmetamode, setvesablank, showconsolefont, showkey, unicode\_start et unicode\_stop

#### Descriptions courtes

<b>chvt</b>	Change le terminal virtuel en avant plan
<b>dealloctv</b>	Désalloue les terminaux virtuels inutilisés
<b>dumpkeys</b>	Affiche les tables de traduction du clavier
<b>fgconsole</b>	Affiche le numéro du terminal virtuel actif
<b>getkeycodes</b>	Affiche la table de correspondance des « scancode » avec les « keycode »
<b>getunimap</b>	Affiche la table de correspondance unicode vers police en cours d'utilisation
<b>kbd_mode</b>	Affiche ou initialise le mode du clavier
<b>kbdrate</b>	Initialise les taux de répétition et de délai du clavier
<b>loadkeys</b>	Charge les tables de traduction du clavier
<b>loadunimap</b>	Charge la table de correspondance entre unicode et police

<b>mapscrn</b>	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 <b>setfont</b>
<b>openvt</b>	Lance un programme sur un nouveau terminal virtuel (VT)
<b>psfaddtable</b>	Un lien vers <b>psfxtable</b>
<b>psfgettable</b>	Un lien vers <b>psfxtable</b>
<b>psfstriptime</b>	Un lien vers <b>psfxtable</b>
<b>psfxtable</b>	Gère les tables de caractères Unicode pour les polices de la console
<b>resizecons</b>	Change l'idée du noyau sur la taille de la console
<b>setfont</b>	Modifie les polices EGA/VGA sur la console
<b>setkeycodes</b>	Charge les entrées de la table de correspondance entre scancode et keycode, utile si vous avez des touches inhabituelles sur votre clavier
<b>setleds</b>	Initialise les drapeaux et LED du clavier
<b>setlogcons</b>	Envoie les messages du noyau sur la console
<b>setmetamode</b>	Définit la gestion des touches meta du clavier
<b>setvesablank</b>	Laisse l'utilisateur configurer la sauvegarde d'écran matérielle (un simple écran vide)
<b>showconsolefont</b>	Affiche la police de l'écran pour la console EGA/VGA
<b>showkey</b>	Affiche les scancodes, keycodes et codes ASCII des touches appuyées sur le clavier
<b>unicode_start</b>	Met le clavier et la console en mode UNICODE. Ne l'utilisez jamais sur LFS car les applications ne sont pas configurées pour supporter UNICODE.
<b>unicode_stop</b>	Remplace le clavier et la console dans le précédent mode UNICODE



## 6.43. E2fsprogs-1.37

Le paquet E2fsprogs contient les outils de gestion du système de fichiers ext2. Il supporte aussi le système de fichiers journalisé ext3.

**Temps de construction estimé :** 0,6 SBU

**Espace disque requis :** 40,0 Mo

**Dépendances de l'installation :** Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Gettext, Glibc, Grep, Make, Sed et Texinfo

### 6.43.1. Installation d'E2fsprogs

Corrigez une erreur de compilation dans la suite de tests d'E2fsprogs :

```
sed -i -e 's/-DTEST/$(ALL_CFLAGS) &/' lib/e2p/Makefile.in
```

Il est recommandé de construire E2fsprogs dans un sous-répertoire du répertoire source :

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

Préparez la compilation d'E2fsprogs :

```
../configure --prefix=/usr --with-root-prefix="" \
  --enable-elf-shlibs --disable-evms
```

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.

`--disable-evms`

Ceci désactive la construction du plugin EVMS (*Enterprise Volume Management System*). Ce plugin n'est pas à jour avec les dernières interfaces internes d'EVMS et EVMS n'est pas installé comme partie intégrante d'un système LFS de base, donc ce plugin n'est pas requis. Voir le site web d'EVMS sur <http://evms.sourceforge.net/> pour plus d'informations concernant EVMS.

Compilez le paquet :

```
make
```

Pour tester les résultats, lancez : **make check**.

Installez les binaires et la documentation :

```
make install
```

Installez aussi les bibliothèques partagées :

```
make install-libs
```

## 6.43.2. Contenu d'E2fsprogs

**Programmes installés:** badblocks, blkid, chattr, compile\_et, debugfs, dumpe2fs, e2fsck, e2image, e2label, findfs, fsck, fsck.ext2, fsck.ext3, logsave, lsattr, mk\_cmds, mke2fs, mkfs.ext2, mkfs.ext3, mklost+found, resize2fs, tune2fs et uuidgen.

**Bibliothèques installées:** libblkid.[a,so], libcom\_err.[a,so], libe2p.[a,so], libext2fs.[a,so], libss.[a,so] et libuuid.[a,so]

### Descriptions courtes

<b>badblocks</b>	Recherche les blocs défectueux sur un périphérique (habituellement une partition d'un disque)
<b>blkid</b>	Un outil en ligne de commande pour trouver et afficher les attributs d'un périphérique bloc
<b>chattr</b>	Modifie les attributs de fichiers sur un système de fichiers ext2 et ext3, la version journalisée d'ext2
<b>compile_et</b>	Un compilateur de table d'erreurs. Il convertit une table de noms d'erreurs et des messages associés en un fichier source C à utiliser avec la bibliothèque com_err library
<b>debugfs</b>	Un débogueur de système de fichiers. Il est utilisé pour examiner et modifier l'état d'un système de fichiers ext2
<b>dumpe2fs</b>	Affiche le superbloc et les informations de groupes de blocs sur le système de fichiers présent sur un périphérique donné
<b>e2fsck</b>	Est utilisé pour vérifier, et quelque fois réparé, les systèmes de fichiers ext2 et ext3
<b>e2image</b>	Est utilisé pour sauver les données critiques d'un système de fichiers ext2 dans un fichier
<b>e2label</b>	Affiche ou modifie le label d'un système de fichiers ext2 présent sur un périphérique donné
<b>findfs</b>	Trouve un système de fichiers par label ou UUID ( <i>Universally Unique Identifier</i> , soit Identifiant Unique Universel)
<b>fsck</b>	Est utilisé pour vérifier, et parfois réparer, les systèmes de fichiers
<b>fsck.ext2</b>	Vérifie par défaut les systèmes de fichiers ext2
<b>fsck.ext3</b>	Vérifie par défaut les systèmes de fichiers ext3
<b>logsave</b>	Sauvegarde la sortie d'une commande dans un journal de traces
<b>lsattr</b>	Liste les attributs de fichiers sur un système de fichiers ext2
<b>mk_cmds</b>	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>libs</code>
<b>mke2fs</b>	Crée un système de fichiers <code>ext2</code> ou <code>ext3</code> sur le périphérique donné
<b>mkfs.ext2</b>	Crée par défaut un système de fichiers <code>ext2</code>
<b>mkfs.ext3</b>	Crée par défaut un système de fichiers <code>ext3</code>
<b>mklost+found</b>	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' <b>e2fsck</b>
<b>resize2fs</b>	est utilisé pour agrandir ou réduire un système de fichiers <code>ext2</code>
<b>tune2fs</b>	Est utilisé pour ajuster les paramètres d'un système de fichiers à condition qu'il soit un <code>ext2</code>
<b>uuidgen</b>	Crée un nouvel UUID. Chaque nouveau 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.
<code>libblkid</code>	Contient des routines pour l'identification de processus et l'extraction de modèles
<code>libcom_err</code>	La routine d'affichage d'erreurs
<code>libe2p</code>	Est utilisé par <b>dumpe2fs</b> , <b>chattr</b> et <b>lsattr</b>
<code>libext2fs</code>	Contient des routines pour permettre aux programmes niveau utilisateur de manipuler un système de fichiers <code>ext2</code>
<code>libs</code>	Utilisé par <b>debugfs</b>
<code>libuuid</code>	Contient des routines pour générer des identifiants uniques pour les objets qui pourraient être accessibles en dehors du système local

## 6.44. Grep-2.5.1a

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

**Temps de construction estimé :** 0,1 SBU

**Espace disque requis :** 4,5 Mo

**Dépendances de l'installation :** Bash, Binutils, Coreutils, Diffutils, GCC, Gettext, Glibc, Make, Sed et Texinfo

### 6.44.1. Installation de Grep

Préparez la compilation de Grep :

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

Compilez le paquet :

```
make
```

Pour tester les résultats, lancez : **make check**.

Installez le paquet :

```
make install
```

### 6.44.2. Contenu de Grep

**Programmes installés:** egrep (lien vers grep), fgrep (lien vers grep) et grep

#### Descriptions courtes

**egrep** Affiche les lignes correspondant à une expression rationnelle étendue

**fgrep** Affiche des lignes correspondant à une liste de chaînes fixes

**grep** Affiche des lignes correspondant à une expression rationnelle basique

## 6.45. Grub-0.96

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

**Temps de construction estimé :** 0,2 SBU

**Espace disque requis :** 10,0 Mo

**Dépendances de l'installation :** Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Grep, Make, Ncurses et Sed

### 6.45.1. Installation de Grub

Ce paquet est connu pour avoir des soucis quand les options d'optimisation par défaut (en incluant les options *-march* et *-mcpu*) sont modifiées. Donc, si des variables d'environnement qui surchargent les optimisations par défaut, telles que *CFLAGS* et *CXXFLAGS*, ont été définies, supprimez cette initialisation pour la construction de Grub.

Préparez la compilation de Grub :

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

Compilez le paquet :

```
make
```

Pour tester les résultats, lancez : **make check**.

Notez que les résultats des tests afficheront toujours l'erreur « *ufs2\_stage1\_5 is too big* ». Ceci est dû à un problème du compilateur mais peut être ignoré sauf si vous planifiez de démarrer à partir d'une partition LFS. Normalement, les partitions sont uniquement utilisées par les stations de travail Sun.

Installez le paquet :

```
make install
mkdir -v /boot/grub
cp -v /usr/lib/grub/i386-pc/stage{1,2} /boot/grub
```

Remplacez *i386-pc* par le répertoire adéquat pour le matériel utilisé.

Le répertoire *i386-pc* contient aussi un certain nombre de fichiers *\*stage1\_5*, différents suivant les différents systèmes de fichiers. Jetez un œil aux fichiers disponibles et copiez les bons dans le répertoire */boot/grub*. La plupart des utilisateurs copieront les fichiers *e2fs\_stage1\_5* et/ou *reiserfs\_stage1\_5*.

### 6.45.2. Contenu de Grub

**Programmes installés:** grub, grub-install, grub-md5-crypt, grub-terminfo et mbchk

#### Descriptions courtes

<b>grub</b>	Le shell de commande pour Grub
<b>grub-install</b>	Installe GRUB sur le périphérique spécifié
<b>grub-md5-crypt</b>	Crypte un mot de passe au format MD5
<b>grub-terminfo</b>	Génère une commande terminfo à partir d'un nom terminfo. Il est utilisable si vous

**mbchk**

avez un terminal non usuel

Vérifie le format d'un noyau multi-boot

## 6.46. Gzip-1.3.5

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

**Temps de construction estimé :** 0,1 SBU

**Espace disque requis :** 2,2 Mo

**Dépendances de l'installation :** Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Grep, Make et Sed

### 6.46.1. Installation de Gzip

Gzip a deux vulnérabilités connues. Le correctif suivant s'occupe des deux problèmes :

```
patch -Np1 -i ../gzip-1.3.5-security_fixes-1.patch
```

Préparez la compilation de Gzip :

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

Le script **gzexe** contient, codé en dur, l'emplacement du binaire **gzip**. Comme l'emplacement de ce binaire changera plus tard, la commande suivante nous assure que le nouvel emplacement sera placé dans le script :

```
sed -i 's@"BINDIR"@"/bin@g' gzexe.in
```

Compilez le paquet :

```
make
```

Installez le paquet :

```
make install
```

Déplacez les programmes dans le répertoire `/bin` et créez quelques liens symboliques couramment utilisés :

```
mv -v /usr/bin/gzip /bin
rm -v /usr/bin/{gunzip,zcat}
ln -sv gzip /bin/gunzip
ln -sv gzip /bin/zcat
ln -sv gzip /bin/compress
ln -sv gunzip /bin/uncompress
```

### 6.46.2. Contenu de Gzip

**Programmes installés:** compress (lien vers gzip), gunzip (lien vers gzip), gzexe, gzip, uncompress (lien vers gunzip), zcat (lien vers gzip), zcmp, zdiff, zegrep, zfgrep, zforce, zgrep, zless, zmore et znew

#### Descriptions courtes

<b>compress</b>	Comprime et déprime des fichiers
<b>gunzip</b>	Déprime les fichiers gzip
<b>gzexe</b>	Crée des fichiers exécutables auto-extractibles
<b>gzip</b>	Comprime les fichiers données en utilisant le codage Lempel-Ziv (LZ77)

<b>uncompress</b>	Décompresse les fichiers compressés
<b>zcat</b>	Décompresse les fichiers gzip sur la sortie standard
<b>zcmp</b>	Lance <b>cmp</b> sur des fichiers compressés avec gzip
<b>zdiff</b>	Lance <b>diff</b> sur des fichiers compressés avec gzip
<b>zegrep</b>	Lance <b>egrep</b> sur des fichiers compressés avec gzip
<b>zfgrep</b>	Lance <b>fgrep</b> sur des fichiers compressés avec gzip
<b>zforce</b>	Force une extension <code>.gz</code> sur tous les fichiers donnés qui sont au format gzip, pour que <b>gzip</b> ne les compresse pas de nouveau ; ceci est utile quand les noms de fichiers sont tronqués lors d'un transfert de fichiers
<b>zgrep</b>	Lance <b>grep</b> sur des fichiers compressés avec gzip
<b>zless</b>	Lance <b>less</b> sur des fichiers compressés avec gzip
<b>zmore</b>	Lance <b>more</b> sur des fichiers compressés avec gzip
<b>znew</b>	Re-compresse des fichiers à partir du format <b>compress</b> vers le format <b>gzip</b> — <code>.Z</code> to <code>.gz</code>



## 6.47. Hotplug-2004\_09\_23

Le paquetage Hotplug contient des scripts qui réagissent aux événements hotplug générés par le noyau. De tels événements correspondent à chaque modification de l'état du noyau, visible dans le système de fichiers `sysfs`, c'est-à-dire l'ajout et la suppression de matériels. Ce paquetage détecte aussi le matériel existant lors du démarrage et insère les modules adéquats dans le noyau en cours d'exécution.

**Temps de construction estimé :** 0,01 SBU

**Espace disque requis :** 460 Ko

**Dépendances de l'installation :** Bash, Coreutils, Find, Gawk et Make

### 6.47.1. Installation de Hotplug

Installez le paquetage Hotplug :

```
make install
```

Copiez un fichier que la cible « install » oublie.

```
cp -v etc/hotplug/pnp.distmap /etc/hotplug
```

Supprimez le script de démarrage qu'Hotplug installe car nous allons utiliser le script inclus dans le paquetage LFS-Bootscripts :

```
rm -rfv /etc/init.d
```

La découverte via Hotplug des périphériques réseau n'est pas encore supportée par le paquetage LFS-Bootscripts. Pour cette raison, supprimez l'agent hotplug pour le réseau :

```
rm -fv /etc/hotplug/net.agent
```

Créez un répertoire pour stocker les *firmware* pouvant être chargés par **hotplug** :

```
mkdir -v /lib/firmware
```

### 6.47.2. Contenu de Hotplug

**Programme installé:** hotplug

**Scripts installés:** /etc/hotplug/\*.rc, /etc/hotplug/\*.agent

**Fichiers installés:** /etc/hotplug/hotplug.functions, /etc/hotplug/blacklist, /etc/hotplug/{pci,usb}, /etc/hotplug/usb.usermap, /etc/hotplug.d et /var/log/hotplug/events

## Descriptions courtes

### **hotplug**

Ce script est appelé par défaut par le noyau Linux lorsque quelque chose modifie son état interne (par exemple, un nouveau périphérique est ajouté ou un périphérique existant est supprimé)

### **/etc/hotplug/\*.rc**

Ces scripts sont utilisés pour le branchement à froid, par exemple détecter et agir avec le matériel déjà présents lors du démarrage du système. Ils sont appelés par le script de démarrage `hotplug` inclus dans le paquetage `LFS-Bootscripts`. Les scripts `*.rc` essaient de récupérer les événements `hotplug` qui ont été perdus lors du démarrage du système parce que, par exemple, le système de fichiers `root` n'a pas été monté par le noyau

### **/etc/hotplug/\*.agent**

Ces scripts sont appelés par **hotplug** en réponse à différents types d'événements `hotplug` générés par le noyau. Leur action revient à insérer les modules correspondant du noyau et à appeler tous scripts fournis par l'utilisateur

### **/etc/hotplug/blacklist**

Ce fichier contient la liste des modules qui ne devraient jamais être insérés dans le noyau par les scripts `Hotplug`

### **/etc/hotplug/hotplug.functions**

Ce fichier contient les fonctions communes utilisées par d'autres scripts dans le paquetage `Hotplug`

### **/etc/hotplug/{pci,usb}**

Ces répertoires contiennent des gestionnaires d'événements `hotplug` écrits par les utilisateurs

### **/etc/hotplug/usb.usermap**

Ce fichier contient des règles pour déterminer quel gestionnaire défini par l'utilisateur appeler pour chaque périphérique USB, suivant l'identifiant du vendeur et d'autres attributs

### **/etc/hotplug.d**

Ce répertoire contient des programmes (ou des liens symboliques vers ces programmes) intéressés pour recevoir des événements `hotplug`. Par exemple, `Udev` place son lien symbolique ici lors de l'installation

### **/lib/firmware**

Ce répertoire contient le firmware pour les périphériques qui ont besoin que leur firmware soit chargé avant d'être utilisable

### **/var/log/hotplug/events**

Ce fichier contient tous les événements que **hotplug** a appelé depuis le démarrage

## 6.48. Man-1.5p

Le paquet Man contient des programmes de recherche et de visualisation de pages man.

**Temps de construction estimé :** 0,1 SBU

**Espace disque requis :** 2,9 Mo

**Dépendances de l'installation :** Bash, Binutils, Coreutils, Gawk, GCC, Glibc, Grep, Make et Sed

### 6.48.1. Installation de Man

Deux ajustements doivent être effectués sur les sources de Man.

Le premier est une substitution **sed** pour ajouter l'option `-R` à la variable `PAGER` de façon à ce que les séquences d'échappement soient correctement gérées par Less :

```
sed -i 's@-is@&R@g' configure
```

La seconde est aussi une substitution **sed** décommentant la ligne « `MANPATH /usr/man` » dans le fichier `man.conf` pour supprimer les résultats redondants lors de l'utilisation de programmes comme **whatis** :

```
sed -i 's@MANPATH./usr/man#@#&@g' src/man.conf.in
```

Préparez la compilation de Man :

```
./configure -confdir=/etc
```

Voici la signification des options de configure :

`-confdir=/etc`

Ceci indique au programme **man** de rechercher le fichier de configuration `man.conf` dans le répertoire `/etc`.

Compilez le paquet :

```
make
```

Installez-le :

```
make install
```



#### Note

Si vous travaillez sur un terminal qui ne supporte pas les attributs de texte comme la couleur ou l'emphase, vous pouvez désactiver les séquences d'échappement « Select Graphic Rendition (SGR) » en modifiant le fichier `man.conf` et en ajoutant l'option `-c` à la variable `NROFF`. Si vous utilisez plusieurs types de terminaux pour un ordinateur, il serait mieux d'ajouter de façon sélective la variable d'environnement `GROFF_NO_SGR` pour les terminaux qui ne supportent pas SGR.

Si l'ensemble de caractères de la locale utilise les caractères 8 bits, recherchez la ligne commençant avec « NROFF » dans `/etc/man.conf` et vérifiez qu'elle correspond à la suivante :

```
NROFF /usr/bin/nroff -Tlatin1 -mandoc
```

Notez que « latin1 » devrait être utilisé même s'il ne s'agit pas de l'ensemble de caractères de la locale. La raison en est que, suivant la spécification, **groff** n'a aucun moyen de présenter les caractères en dehors de l'ISO 8859-1 sans quelques codes d'échappement étranges. En formatant les pages man, **groff** pense qu'elles sont dans le codage ISO 8859-1 et le commutateur `-Tlatin1` indique à **groff** d'utiliser le même codage pour la sortie. Comme **groff** ne recode pas les caractères en entrée, le résultat formaté est réellement dans le même codage qu'en entrée et est, du coup, utilisable en entrée par un paginateur.

Ceci ne résout pas le problème du programme bogué **man2dvi** pour les pages man localisés dans des locales autres que ISO 8859-1. De plus, cela ne fonctionne pas avec des ensembles de caractères multi-octets. Le premier problème n'a pas de solution. Le second n'a pas d'importance car l'installation LFS ne supporte pas les ensembles de caractères multi-octets.

Des informations supplémentaires concernant la compression des pages man et info sont disponibles dans le livre BLFS sur <http://www.linuxfromscratch.org/blfs/view/cvs/postlfs/compressdoc.html>.

## 6.48.2. Contenu de Man

**Programmes installés:** `apropos`, `makewhatis`, `man`, `man2dvi`, `man2html` et `whatis`

### Descriptions courtes

<b>apropos</b>	Cherche dans la base de données <b>whatis</b> et affiche les descriptions courtes des commandes systèmes contenant une chaîne donnée
<b>makewhatis</b>	Construit la base de données <b>whatis</b> ; elle lit toutes les pages man comprises dans <code>MANPATH</code> et, pour chaque page, écrit le nom et une courte description dans la base de données <b>whatis</b>
<b>man</b>	Formate et affiche la page man demandée en ligne
<b>man2dvi</b>	Convertit une page man au format dvi
<b>man2html</b>	Convertit une page man au format HTML
<b>whatis</b>	Recherche dans la base de données <b>whatis</b> et affiche les descriptions courtes des commandes système contenant un mot clé donné comme un mot séparé

## 6.49. Make-3.80

Le paquet Make contient un programme pour compiler des paquetages.

**Temps de construction estimé :** 0,2 SBU

**Espace disque requis :** 7,1 Mo

**Dépendances de l'installation :** Bash, Binutils, Coreutils, Diffutils, GCC, Gettext, Glibc, Grep et Sed

### 6.49.1. Installation de Make

Préparez la compilation de Make :

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

Compilez le paquet :

```
make
```

Pour tester les résultats, lancez : **make check**.

Installez le paquet :

```
make install
```

### 6.49.2. Contenu de Make

**Programme installé:** make

#### Descriptions courtes

**make** Détermine automatiquement quelles pièces d'un paquetage doivent être (re)compilées. Puis, il lance les commandes adéquates

## 6.50. Module-Init-Tools-3.1

Le paquet Module-Init-Tools contient des programmes de gestion des modules des noyaux Linux pour les versions 2.5.47 et ultérieures.

**Temps de construction estimé :** 0,1 SBU

**Espace disque requis :** 4,9 Mo

**Dépendances de l'installation :** Bash, Binutils, Bison, Coreutils, Diffutils, Flex, GCC, Glibc, Grep, M4, Make et Sed

### 6.50.1. Installation de Module-Init-Tools

Module-Init-Tools tente de ré-écrire sa page `man modprobe.conf` lors de la construction. Ceci n'est pas nécessaire et repose aussi sur la commande **docbook2man** — qui n'est pas installé dans LFS. Lancez la commande suivante pour éviter ceci :

```
touch modprobe.conf.5
```

Si vous souhaitez exécuter la suite de tests du Module-Init-Tools, vous aurez besoin de télécharger l'archive tar séparé. Exécutez les commandes suivantes pour réaliser les tests (notez que la commande **make distclean** est requise pour nettoyer le répertoire des sources car les sources sont de nouveau compilés par le processus de test) :

```
tar -xvf ../module-init-tools-testsuite-3.1.tar.bz2 --strip-path=1 &&
./configure &&
make check &&
make distclean
```

Préparez la compilation de Module-Init-Tools :

```
./configure --prefix="" --enable-zlib
```

Voici la signification des options de configure :

`--enable-zlib`

Ceci permet au paquetage Module-Init-Tools de gérer les modules noyau compressés.

Compilez le paquet :

```
make
```

Installez le paquet :

```
make install
```

### 6.50.2. Contenu de Module-Init-Tools

**Programmes installés:** `depmod`, `insmod`, `insmod.static`, `lsmod` (lien vers `insmod`), `modinfo`, `modprobe` (lien vers `insmod`) et `rmmod` (lien vers `insmod`)

#### Descriptions courtes

**depmod**                    Crée un fichier de dépendances basé sur les symboles trouvés dans l'ensemble de

modules existants ; ce fichier de dépendances est utilisé par **modprobe** pour charger automatiquement les modules requis

<b>insmod</b>	Installe un module chargeable dans le noyau en cours d'exécution
<b>insmod.static</b>	Une version compilée statiquement de <b>insmod</b>
<b>lsmod</b>	Liste les modules déjà chargés
<b>modinfo</b>	Examine un fichier objet associé à un module du noyau et affiche toute information qu'il peut récupérer
<b>modprobe</b>	Utilise un fichier de dépendances, créé par <b>depmod</b> , pour charger automatiquement les modules adéquats
<b>rmmod</b>	Décharge les modules du noyau en cours d'exécution

## 6.51. Patch-2.5.4

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

**Temps de construction estimé** : 0,1 SBU

**Espace disque requis** : 1,5 Mo

**Dépendances de l'installation** : Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Grep, Make et Sed

### 6.51.1. Installation de Patch

Préparez la compilation de Patch. Le commutateur `-D_GNU_SOURCE` du préprocesseur est seulement nécessaire pour les PowerPC. Pour les autres machines, il est inutile) :

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

Compilez le paquet :

```
make
```

Ce paquet ne fournit pas de suite de tests.

Installez le paquet :

```
make install
```

### 6.51.2. 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.



## 6.52. Procps-3.2.5

Le paquet Procps contient des programmes pour surveiller les processus.

**Temps de construction estimé :** 0,1 SBU

**Espace disque requis :** 2,3 Mo

**Dépendances de l'installation :** Bash, Binutils, Coreutils, GCC, Glibc, Make et Ncurses

### 6.52.1. Installation de Procps

Compilez le paquet :

```
make
```

Installez le paquet :

```
make install
```

### 6.52.2. Contenu de Procps

**Programmes installés:** free, kill, pgrep, pkill, pmap, ps, skill, snice, sysctl, tload, top, uptime, vmstat, w et watch

**Bibliothèque installée:** libproc.so

#### Descriptions courtes

<b>free</b>	Indique le total de mémoire libre et utilisé sur le système à la fois pour la mémoire physique et pour la mémoire swap
<b>kill</b>	Envoie des signaux aux processus
<b>pgrep</b>	Recherche les processus suivant leur nom et autres attributs
<b>pkill</b>	Envoie des signaux aux processus suivant leur nom et autres attributs
<b>pmap</b>	Affiche le plan mémoire du processus désigné
<b>ps</b>	Donne un aperçu des processus en cours d'exécution
<b>skill</b>	Envoie des signaux aux processus correspondant à un critère donné
<b>snice</b>	Modifie les priorités des processus suivant le critère donné
<b>sysctl</b>	Modifie les paramètres du noyau en cours d'exécution
<b>tload</b>	Affiche un graphe de la charge système actuelle
<b>top</b>	Affiche une liste des processus demandant le maximum de ressources CPU ; il fournit un affichage agréable sur l'activité du processeur en temps réel
<b>uptime</b>	Affiche le temps d'exécution du système, le nombre d'utilisateurs connectés et les moyennes de charge système

- vmstat** Affiche les statistiques de mémoire virtuelle, donne des informations sur les processus, la mémoire, la pagination, le nombre de blocs en entrées/sorties, les échappements et l'activité CPU
- w** Affiche les utilisateurs actuellement connectés, où et depuis quand
- watch** Lance une commande de manière répétée, affichant le premier écran de sa sortie ; ceci vous permet de surveiller la sortie
- libproc** Contient les fonctions utilisées par la plupart des programmes de ce paquet

## 6.53. Psmisc-21.6

Le paquet Psmisc contient des programmes pour afficher des informations sur les processus en cours d'exécution.

**Temps de construction estimé :** 0,1 SBU

**Espace disque requis :** 1,7 Mo

**Dépendances de l'installation :** Bash, Binutils, Coreutils, Diffutils, GCC, Gettext, Glibc, Grep, Make, Ncurses et Sed

### 6.53.1. Installation de Psmisc

Préparez la compilation de Psmisc pour :

```
./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 certaines de binaires de Psmisc sont utilisés dans des scripts de démarrage.

Compilez le paquet :

```
make
```

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
```

### 6.53.2. Contenu de Psmisc

**Programmes installés:** `fuser`, `killall`, `pstree` et `pstree.x11` (lien vers `pstree`)

#### Descriptions courtes

<b>fuser</b>	Indique les PID de processus utilisant les fichiers ou systèmes de fichiers donnés
<b>killall</b>	Tue les processus suivant leur nom. Il envoie un signal à tous les processus en cours
<b>pstree</b>	Affiche les processus en cours hiérarchiquement
<b>pstree.x11</b>	Identique à <b>pstree</b> , si ce n'est qu'il attend une confirmation avant de quitter

## 6.54. Shadow-4.0.9

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

**Temps de construction estimé :** 0,4 SBU

**Espace disque requis :** 13,7 Mo

**Dépendances de l'installation :** Bash, Binutils, Bison, Coreutils, Diffutils, GCC, Gettext, Glibc, Grep, Make et Sed

### 6.54.1. Installation de Shadow



#### Note

Si vous voulez forcer l'utilisation de mots de passe forts, référez-vous à <http://www.linuxfromscratch.org/blfs/view/svn/postlfs/cracklib.html> pour l'installation de Cracklib avant de construire Shadow. Puis, ajoutez `--with-libcrack` à la commande **configure** ci-dessous.

Préparez la compilation de Shadow :

```
./configure --libdir=/lib --enable-shared
```

Désactivez l'installation du programme **groups** et de sa page man car Coreutils fournit une meilleure version :

```
sed -i 's/groups$(EXEEXT) //' src/Makefile
sed -i '/groups/d' man/Makefile
```

Compilez le paquet :

```
make
```

Installez le paquet :

```
make install
```

Shadow utilise deux fichiers pour configurer les paramètres d'authentification du système. Installez ces deux fichiers de configuration :

```
cp -v etc/{limits,login.access} /etc
```

Au lieu d'utiliser la méthode *crypt* par défaut, nous voulons utiliser la méthode *MD5* plus sécurisée pour le cryptage des mots de passe, qui autorise aussi une taille de mot de passe de plus de huit caractères. Il est aussi nécessaire de modifier l'emplacement obsolète `/var/spool/mail` des boîtes de courrier électronique des utilisateurs pour que Shadow utilise par défaut le nouveau `/var/mail` disponible maintenant. Ces deux points se réalisent en modifiant le fichier de configuration adéquat tout en le copiant à sa destination :



#### Note

Si vous construisez Shadow avec le support de Cracklib, insérez ce qui suit dans la commande **sed** indiquée ci-dessous :

```
-e 's@DICTPATH.*@DICTPATH\t/lib/cracklib/pw_dict@'
```

```
sed -e 's#@MD5_CRYPT_ENAB.no@MD5_CRYPT_ENAB yes@' \  
-e 's@/var/spool/mail@/var/mail@' \  
etc/login.defs.linux > /etc/login.defs
```

Déplacez un programme au bon emplacement :

```
mv -v /usr/bin/passwd /bin
```

Déplacez les bibliothèques de Shadow dans des emplacements plus appropriés :

```
mv -v /lib/libshadow.*a /usr/lib  
rm -v /lib/libshadow.so  
ln -sfv ../../lib/libshadow.so.0 /usr/lib/libshadow.so
```

L'option `-D` du programme `useradd` requiert le répertoire `/etc/default` pour fonctionner correctement :

```
mkdir -v /etc/default
```

## 6.54.2. 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 *compatible avec shadow*, c'est-à-dire qu'ils ont besoin d'être capables de fonctionner avec des mots de passe shadow.

Pour activer les mots de passe shadow, lancez la commande suivante :

```
pwconv
```

Pour activer les mots de passe shadow pour les groupes, lancez :

```
grpconv
```

Sous des circonstances normales, vous n'avez pas encore créé de mots de passe. Néanmoins, si vous revenez plus tard dans cette section pour activer les mots de passe shadow, réinitialisez tous les mots de passe des utilisateurs actuels avec la commande `passwd` et les mots de passe des groupes avec la commande `gpasswd`.

## 6.54.3. Configurer le mot de passe de root

Choisissez un mot de passe pour l'utilisateur `root` et configurez-le avec :

```
passwd root
```

## 6.54.4. Contenu de Shadow

**Programmes installés:** `chage`, `chfn`, `chpasswd`, `chsh`, `expiry`, `faillog`, `gpasswd`, `groupadd`, `groupdel`, `groupmod`, `grpck`, `grpconv`, `grpunconv`, `lastlog`, `login`, `logoutd`, `mkpasswd`, `newgrp`, `newusers`, `passwd`, `pwck`, `pwconv`, `pwunconv`, `sg` (lien vers `newgrp`), `useradd`, `userdel`, `usermod`, `vigr` (lien vers `vipw`) et `vipw`

**Bibliothèques installées:** `libshadow.[a,so]`

### Descriptions courtes

<b>chage</b>	Utilisé pour modifier le nombre maximum de jours avant d'obliger un changement de mot de passe
<b>chfn</b>	Utilisé pour modifier le nom complet de l'utilisateur et quelques autres informations
<b>chpasswd</b>	Utilisé pour mettre à jour les mots de passe d'une série de compte en une fois
<b>chsh</b>	Utilisé pour modifier le shell de connexion par défaut d'un utilisateur
<b>expiry</b>	vérifie et renforce la politique d'expiration des mots de passe
<b>faillog</b>	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
<b>gpsswd</b>	Est utilisé pour ajouter et supprimer des membres et des administrateurs aux groupes
<b>groupadd</b>	Crée un groupe avec le nom donné
<b>groupdel</b>	Supprime le groupe ayant le nom donné
<b>groupmod</b>	Est utilisé pour modifier le nom ou le GID du groupe
<b>grpck</b>	Vérifie l'intégrité des fichiers <code>/etc/group</code> et <code>/etc/gshadow</code>
<b>grpconv</b>	Crée ou met à jour le fichier shadow à partir du fichier group standard
<b>grpunconv</b>	Met à jour <code>/etc/group</code> à partir de <code>/etc/gshadow</code> puis supprime ce dernier
<b>lastlog</b>	Indique les connexions les plus récentes de tous les utilisateurs ou d'un utilisateur donné
<b>login</b>	Est utilisé par le système pour permettre aux utilisateurs de se connecter
<b>logoutd</b>	Est un démon utilisé pour renforcer les restrictions sur les temps et ports de connexion
<b>mkpasswd</b>	Crypte les mots de passe au hasard
<b>newgrp</b>	Est utilisé pour modifier le GID courant pendant une session de connexion
<b>newusers</b>	Est utilisé pour créer ou mettre à jour toute une série de comptes utilisateur en une fois
<b>passwd</b>	Est utilisé pour modifier le mot de passe d'un utilisateur ou d'un groupe
<b>pwck</b>	Vérifie l'intégrité des fichiers de mots de passe, <code>/etc/passwd</code> et <code>/etc/shadow</code>
<b>pwconv</b>	Crée ou met à jour le fichier de mots de passe shadow à partir du fichier password habituel
<b>pwunconv</b>	Met à jour <code>/etc/passwd</code> à partir de <code>/etc/shadow</code> puis supprime ce dernier
<b>sg</b>	Exécute une commande donnée lors de l'initialisation du GID de l'utilisateur à un groupe donné
<b>su</b>	Lance un shell en substituant les ID de l'utilisateur et du groupe
<b>useradd</b>	Crée un nouvel utilisateur avec le nom donné ou met à jour les informations par défaut du nouvel utilisateur
<b>userdel</b>	Supprime le compte utilisateur indiqué
<b>usermod</b>	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
<b>vigr</b>	Édite les fichiers <code>/etc/group</code> et <code>/etc/gshadow</code>
<b>vipw</b>	Édite les fichiers <code>/etc/passwd</code> et <code>/etc/shadow</code>
<b>libshadow</b>	Contient des fonctions utilisées par la plupart des programmes de ce paquet

## 6.55. Sysklogd-1.4.1

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

**Temps de construction estimé :** 0,1 SBU

**Espace disque requis :** 0,5 Mo

**L'installation de Sysklogd dépend de:** Binutils, Coreutils, GCC, Glibc et Make

### 6.55.1. Installation de Sysklogd

Sysklogd a quelques soucis avec la série 2.6 du noyau Linux. Corrigez-les en appliquant le correctif suivant :

```
patch -Np1 -i ../sysklogd-1.4.1-fixes-1.patch
```

Compilez le paquet :

```
make
```

Installez le paquet :

```
make install
```



## 6.55.2. Configurer Syslogd

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

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

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

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

## 6.55.3. Contenu de Syslogd

**Programmes installés:** klogd et syslogd

### Descriptions courtes

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

**syslogd**     Trace les messages que les programmes systèmes donnent

## 6.56. Sysvinit-2.86

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

**Temps de construction estimé :** 0,1 SBU

**Espace disque requis :** 1012 Ko

**Dépendances de l'installation :** Binutils, Coreutils, GCC, Glibc et Make

### 6.56.1. Installation de Sysvinit

Lorsque les niveaux d'exécution changent (par exemple, lors de l'arrêt du système), **init** envoie des signaux de fin aux processus qu'**init** a lui-même lancé et qui ne devraient plus s'exécuter dans le nouveau niveau d'exécution. En faisant ceci, **init** affiche des messages comme « Sending processes the TERM signal » (NdT : Envoi du signal TERM aux processus) ce qui semble impliquer qu'il envoie ce signal à tous les processus en cours d'exécution. Pour éviter cette mauvaise interprétation, modifiez les sources pour que ce message soit remplacé par « Sending processes started by init the TERM signal » (NdT : Envoi du signal TERM aux processus lancés par init) :

```
sed -i 's@Sending processes@& started by init@g' \
    src/init.c
```

Compilez le paquet :

```
make -C src
```

Puis, installez le paquet :

```
make -C src install
```

### 6.56.2. Configurer Sysvinit

Créez un nouveau fichier `/etc/inittab` en lançant ce qui suit :

```
cat > /etc/inittab << "EOF"
# Début /etc/inittab

id:3:initdefault:

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

l0:0:wait:/etc/rc.d/init.d/rc 0
l1:S1:wait:/etc/rc.d/init.d/rc 1
l2:2:wait:/etc/rc.d/init.d/rc 2
l3:3:wait:/etc/rc.d/init.d/rc 3
l4:4:wait:/etc/rc.d/init.d/rc 4
l5:5:wait:/etc/rc.d/init.d/rc 5
l6:6:wait:/etc/rc.d/init.d/rc 6

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

su:S016:once:/sbin/sulogin

1:2345:respawn:/sbin/agetty -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

# Fin /etc/inittab
EOF
```

L'option `-I '\033(K'` indique à **agetty** d'envoyer cette séquence d'échappement au terminal avant de faire quoi que ce soit d'autres. Cette séquence d'échappement bascule l'ensemble de caractères de la console sur un défini par l'utilisateur, qui peut être modifié en exécutant le programme **setfont**. Le script de démarrage **console** du paquet LFS-Bootscripts appelle le programme **setfont** au lancement du système. L'envoi de cette séquence d'échappement est nécessaire pour les personnes utilisant des polices autres qu'ISO 8859-1 mais n'a aucun effet sur les personnes de langue anglaise.

### 6.56.3. Contenu de Sysvinit

**Programmes installés:** halt, init, killall5, last, lastb (lien vers last), mesg, pidof (lien vers killall5), poweroff (lien vers halt), reboot (lien vers halt), runlevel, shutdown, sulogin, telinit (lien vers init), utmpdump et wall

#### Descriptions courtes

<b>halt</b>	Lance normalement <b>shutdown</b> 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
<b>init</b>	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
<b>killall5</b>	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
<b>last</b>	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
<b>lastb</b>	Affiche les tentatives échouées de connexions tracées dans <code>/var/log/btmp</code>
<b>mesg</b>	Contrôle si les autres utilisateurs peuvent envoyer des messages au terminal de l'utilisateur courant
<b>mountpoint</b>	Vérifie si le répertoire est un point de montage
<b>pidof</b>	Indique le PID des programmes précisés
<b>poweroff</b>	Indique au noyau d'arrêter le système et de couper l'ordinateur (voir <b>halt</b> )
<b>reboot</b>	Indique au noyau de redémarrer le système (voir <b>halt</b> )
<b>runlevel</b>	Indique le niveau d'exécution actuel et précédent comme précisé dans l'enregistrement du dernier niveau d'exécution dans <code>/var/run/utmp</code>
<b>shutdown</b>	Arrête proprement le système en le signalant à tous les processus et à tous les utilisateur connectés
<b>sulogin</b>	Permet la connexion du superutilisateur. Il est normalement appelé par <b>init</b> lorsque le système passe en mono-utilisateur
<b>telinit</b>	indique à <b>init</b> dans quel niveau d'exécution entrer
<b>utmpdump</b>	Affiche le contenu du fichier de connexion donné dans un format plus agréable
<b>wall</b>	Écrit un message à l'intention de tous les utilisateurs connectés

## 6.57. Tar-1.15.1

Le paquet Tar contient un programme d'archivage.

**Temps de construction estimé :** 0,2 SBU

**Espace disque requis :** 12,7 Mo

**Dépendances de l'installation :** Bash, Binutils, Coreutils, Diffutils, GCC, Gettext, Glibc, Grep, Make et Sed

### 6.57.1. Installation de Tar

Tar a un bogue quand l'option `-S` est utilisée avec des fichiers de plus de 4 Go. Le correctif suivant corrige proprement ce problème :

```
patch -Np1 -i ../tar-1.15.1-sparse_fix-1.patch
```

Préparez la compilation de Tar :

```
./configure --prefix=/usr --bindir=/bin --libexecdir=/usr/sbin
```

Compilez le paquet :

```
make
```

Pour tester les résultats, lancez : `make check`.

Installez le paquet :

```
make install
```

### 6.57.2. Contenu de Tar

**Programmes installés:** rmt et 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

## 6.58. Udev-056

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

**Temps de construction estimé :** 0,1 SBU

**Espace disque requis :** 6,7 Mo

**Dépendances de l'installation :** Coreutils et Make

### 6.58.1. Installation d'Udev

Compilez le paquet :

```
make udevdir=/dev
```

```
udevdir=/dev
```

Ceci indique à **udev** le répertoire où les nœuds périphériques doivent être créés.

Pour tester les résultats, exécutez : **make test**.

Installez le paquet :

```
make DESTDIR=/ udevdir=/dev install
```

Voici la signification de l'option de make :

```
DESTDIR=/
```

Ceci empêche le processus de construction d'Udev de tuer tous les processus **udev** qui pourraient être en cours d'exécution sur le système hôte.

La configuration par défaut d'Udev est loin d'être idéale, donc installez les fichiers de configuration maintenant :

```
cp -v ../udev-config-4.rules /etc/udev/rules.d/25-lfs.rules
```

Exécutez le programme **udevstart** pour créer notre complément des nœuds de périphériques.

```
/sbin/udevstart
```

### 6.58.2. Contenu d'Udev

**Programmes installés:** udev, udevd, udevsend, udevstart, udevinfo et udevtest

**Répertoire installé:** /etc/udev

#### Descriptions courtes

**udev**            Crée les nœuds périphériques dans /dev ou renomme les interfaces réseau (pas dans LFS) en réponse aux événements de montage à chaud

<b>udev</b>	Un démon qui réordonne les événements de montage à chaud avant de les soumettre à <b>udev</b> , évitant ainsi différentes conditions particulières
<b>udevsend</b>	Délivre les événements de montage à chaud à <b>udev</b>
<b>udevstart</b>	Crée des nœuds périphériques dans <code>/dev</code> correspondant aux pilotes compilés directement dans le noyau ; il réalise des tâches en simulant les événements de montage à chaud présumément supprimés par le noyau avant l'appel de ce programme (parce que le système de fichiers root n'a pas été démonté) et en soumettant des événements synthétiques de montage à chaud à <b>udev</b>
<b>udevinfo</b>	Autorise les utilisateurs à envoyer des requêtes à la base de données <b>udev</b> pour des informations sur tout périphérique actuellement présent sur le système ; il fournit aussi une façon d'envoyer une requête à tout périphérique dans la hiérarchie <code>sysfs</code> pour aider à la création des règles udev
<b>udevtest</b>	Simule un lancement d' <b>udev</b> pour le périphérique donné et affiche le nom du nœud que le vrai <b>udev</b> aurait créé ou (pas dans LFS) le nom de l'interface réseau renommé
<code>/etc/udev</code>	Contient les fichiers de configuration d' <b>udev</b> , les droits des périphériques et les règles du nommage des périphériques

## 6.59. Util-linux-2.12q

Le paquet Util-linux contient différents outils. Parmi eux se trouvent des outils de gestion des systèmes de fichiers, de consoles, de partitions et des messages.

**Temps de construction estimé :** 0,2 SBU

**Espace disque requis :** 11,6 Mo

**Dépendances de l'installation :** Bash, Binutils, Coreutils, Diffutils, GCC, Gettext, Glibc, Grep, Make, Ncurses, Sed et Zlib

### 6.59.1. Notes de compatibilité FHS

Le FHS recommande d'utiliser `/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 's@etc/adjtime@var/lib/hwclock/adjtime@g' \
    hwclock/hwclock.c
mkdir -p /var/lib/hwclock
```

### 6.59.2. Installation d'Util-linux

Util-linux échoue lors de sa compilation avec les nouvelles versions de Linux-Libc-Headers. Le correctif suivant corrige proprement ce problème :

```
patch -Np1 -i ../util-linux-2.12q-cramfs-1.patch
```

Util-linux a une faille de sécurité qui pourrait permettre à un utilisateur de remonter un volume sans l'option `nosuid`. Le correctif suivant s'occupe de ce problème :

```
patch -Np1 -i ../util-linux-2.12q-umount_fix-1.patch
```

Préparez la compilation d'Util-linux :

```
./configure
```

Compilez le paquet :

```
make HAVE_KILL=yes HAVE_SLN=yes
```

Voici la signification des paramètres de make :

*HAVE\_KILL=yes*

Ceci empêche le programme **kill** (déjà installé par Procps) d'être construit et installé de nouveau.

*HAVE\_SLN=yes*

Ceci empêche le programme **sln** (un **ln** lié statiquement déjà installé par Glibc) d'être construit et installé de nouveau.

Ce paquetage ne vient pas avec une suite de tests.

Installez le paquetage et déplacez le binaire **logger** dans `/bin` car il est nécessaire pour le paquetage LFS-Bootscripts :

```
make HAVE_KILL=yes HAVE_SLN=yes install
mv /usr/bin/logger /bin
```



### 6.59.3. Contenu d'Util-linux

**Programmes installés:** agetty, arch, blockdev, cal, cfdisk, chkdupexe, col, colcrt, colrm, column, ctrlaltdel, cytune, ddate, dmesg, elvtune, fdformat, fdisk, fsck.cramfs, fsck.minix, getopt, hexdump, hwclock, ipcrm, ipcs, isosize, line, logger, look, losetup, mcookie, mkfs, mkfs.bfs, mkfs.cramfs, mkfs.minix, mkswap, more, mount, namei, pg, pivot\_root, ramsize (lien vers rdev), raw, rdev, readprofile, rename, renice, rev, rootflags (lien vers rdev), script, setfdprm, setuid, setterm, sfdisk, swapdev, swapoff (lien vers swapon), swapon, tunelp, ul, umount, vidmode (lien vers rdev), whereis et write

#### Descriptions courtes

<b>agetty</b>	Ouvre un port tty, demande un nom de connexion puis appelle le programme <b>login</b>
<b>arch</b>	Affiche l'architecture de la machine
<b>blockdev</b>	Permet aux utilisateurs d'appeler les ioctl d'un périphérique bloc à partir de la ligne de commande
<b>cal</b>	Affiche un calendrier simple
<b>cfdisk</b>	Manipule la table des partitions du périphérique donné
<b>chkdupexe</b>	Trouve les exécutable dupliqués
<b>col</b>	Filtre les retours chariot inversés
<b>colcrt</b>	filtrer la sortie de <b>nroff</b> pour les terminaux manquant de capacités comme le texte barré ou les demi-lignes
<b>colrm</b>	Filtre les colonnes données
<b>column</b>	Formate un fichier donné en de nombreuses colonnes
<b>ctrlaltdel</b>	Initialise la combinaison des touches Ctrl+Alt+Del pour une réinitialisation matérielle ou logicielle
<b>cytune</b>	Est utilisé pour paramétrer finement les pilotes de lignes séries des cartes Cyclades
<b>ddate</b>	Donne la date discordienne ou convertit la date grégorienne en une date discordienne
<b>dmesg</b>	Affiche les messages du noyau lors du démarrage
<b>elvtune</b>	Est utilisé pour configurer finement les performances et l'interactivité d'un périphérique bloc
<b>fdformat</b>	Réalise un formatage de bas niveau sur une disquette
<b>fdisk</b>	Est utilisé pour manipuler la table de partitions du périphérique donné
<b>fsck.cramfs</b>	Réalise un test de consistance sur le système de fichiers Cramfs du périphérique donné
<b>fsck.minix</b>	Réalise un test de consistance sur le système de fichiers Minix du périphérique donné
<b>getopt</b>	Analyse les options sur la ligne de commande donnée
<b>hexdump</b>	Affiche le fichier indiqué en hexadécimal ou dans un autre format donné
<b>hwclock</b>	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> )
<b>ipcrm</b>	Supprime la ressource IPC donnée

<b>ipcs</b>	Fournit l'information de statut IPC
<b>isosize</b>	Affiche la taille d'un système de fichiers iso9660
<b>line</b>	Copie une simple ligne
<b>logger</b>	Enregistre le message donné dans les traces système
<b>look</b>	Affiche les lignes commençant avec la chaîne donnée
<b>losetup</b>	Initialiser et contrôle les périphériques loop
<b>mcookie</b>	génère des cookies magiques, nombres hexadécimaux aléatoires sur 128 bits, pour <b>xauth</b>
<b>mkfs</b>	Construit un système de fichiers sur un périphérique (habituellement une partition du disque dur)
<b>mkfs.bfs</b>	Crée un système de fichiers bfs de SCO (Santa Cruz Operations)
<b>mkfs.cramfs</b>	Crée un système de fichiers cramfs
<b>mkfs.minix</b>	Crée un système de fichiers Minix
<b>mkswap</b>	initialise le périphérique ou le fichier à utiliser comme swap
<b>more</b>	est un filtre pour visualiser un texte un écran à la fois
<b>mount</b>	Attache le système de fichiers du périphérique donné sur un répertoire spécifié dans le système de fichiers
<b>namei</b>	Affiche les liens symboliques dans les chemins donnés
<b>pg</b>	Affiche un fichier texte un écran à la fois
<b>pivot_root</b>	Fait en sorte que le système de fichiers donné soit le nouveau système de fichiers racine du processus actuel
<b>ramsize</b>	Initialise la taille du disque RAM dans une image amorçable
<b>raw</b>	Utilisé pour envoyer une requête et initialiser le périphérique racine et d'autres choses dans une image amorçable
<b>rdev</b>	Utilisé pour envoyer une requête et initialiser le périphérique racine et d'autres choses dans une image amorçable
<b>readprofile</b>	Lit les informations de profilage du noyau
<b>rename</b>	Renomme les fichiers donnés, remplaçant une chaîne donnée par une autre
<b>renice</b>	Modifie la priorité des processus exécutés
<b>rev</b>	Inverse les lignes d'un fichier donné
<b>rootflags</b>	Initialise les options racine d'une image amorçable
<b>script</b>	Crée un script à partir d'une session du terminal, de tout ce qui est affiché sur un terminal
<b>setfdprm</b>	Initialise les paramètres de disquette fournis par l'utilisateur
<b>setsid</b>	Lance le programme donné dans une nouvelle session
<b>setterm</b>	Initialise les attributs du terminal

<b>sfdisk</b>	Est un manipulateur de table de partitions disque
<b>swapdev</b>	Initialise le périphérique swap dans une image amorçable
<b>swapoff</b>	Désactive les périphériques et fichiers de pagination et de swap
<b>swapon</b>	Active les périphériques et fichiers de pagination et de swap, et liste les périphériques et fichiers en cours d'utilisation.
<b>tunelp</b>	Est utilisé pour paramétrer finement une imprimante ligne
<b>ul</b>	Un filtre pour traduire les soulignements en séquences d'échappement indiquant un soulignement pour le terminal utilisé
<b>umount</b>	Déconnecte un système de fichiers à partir de la hiérarchie de fichiers du système
<b>vidmode</b>	Initialise le mode vidéo d'une image amorçable
<b>whereis</b>	Affiche l'emplacement du binaire, les sources et la page de manuel de la commande donnée
<b>write</b>	Envoie un message à l'utilisateur donné <i>sauf si</i> l'utilisateur a désactivé de tels messages

## 6.60. Symboles de débogage

La plupart des programmes et des bibliothèques sont compilés, par défaut, en incluant les symboles de débogage (avec l'option `-g` de `gcc`). Ceci signifie que, lors du débogage d'un programme ou d'une bibliothèque compilé avec les informations de débogage, le débogueur peut vous donner non seulement les adresses mémoire mais aussi les noms des routines et variables.

Néanmoins, l'intégration de ces symboles de débogage font grossir le programme ou la bibliothèque de façon significative. Ce qui suit est un exemple de l'espace occupé par ces symboles :

- un binaire bash avec les symboles de débogage : 1200 Ko
- un binaire bash sans les symboles de débogage : 480 Ko
- les fichiers Glibc et GCC (`/lib` et `/usr/lib`) avec les symboles de débogage : 87 Mo
- les fichiers Glibc et GCC sans les symboles de débogage : 16 Mo

Les tailles peuvent varier suivant le compilateur et la bibliothèque C utilisés mais, lors d'une comparaison de programmes avec et sans symboles de débogages, la différence sera généralement d'un facteur de deux à cinq.

Comme la plupart des gens n'utiliseront jamais un débogueur sur leur système, beaucoup d'espace disque peut être gagné en supprimant ces symboles. La prochaine section montre comment supprimer tous les symboles de débogage des programmes et bibliothèques. Des informations supplémentaires sur l'optimisation du système sont disponibles sous forme d'astuce sur <http://www.linuxfromscratch.org/hints/downloads/files/optimization.txt>.

## 6.61. Supprimer de nouveau les symboles des fichiers objets

Si l'utilisateur initial n'est pas un développeur et ne pense pas faire de débogage sur les logiciels du système, la taille du système peut être diminuée d'environ 200 Mo en supprimant les symboles de débogage contenus dans les binaires et dans les bibliothèques. Ceci ne pose pas de problème autre que le fait de ne plus pouvoir les déboguer.

La plupart des personnes qui utilisent la commande mentionnée ci-dessous ne rencontrent aucune difficulté. Néanmoins, il est facile de faire une erreur de saisie et rendre le nouveau système complètement inutilisable, donc avant d'exécuter la commande **strip**, il est recommandé de faire une sauvegarde de l'état actuel.

Avant d'exécuter la suppression de ces symboles, faites particulièrement attention qu'aucun des binaires concernés ne sont en cours d'exécution. Si vous n'êtes pas sûr que l'utilisateur est entré dans chroot avec la commande donnée dans Section 6.3, « Entrer dans l'environnement chroot, » quittez le chroot :

```
logout
```

Puis, retournez-y :

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

Maintenant, les binaires et les bibliothèques peuvent être traités en toute sécurité :

```
/tools/bin/find /{,usr/}{bin,lib,sbin} -type f \
    -exec /tools/bin/strip --strip-debug '{} ' ;'
```

Un grand nombre de fichiers seront rapportés comme ayant un format non reconnu. Ces messages d'avertissement indiquent que ces fichiers sont des scripts et non pas des binaires.

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.

## 6.62. Nettoyer

À partir de maintenant, en rentrant dans l'environnement chroot après l'avoir quitté, utilisez la commande chroot suivante :

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

La raison en est que les programmes dans `/tools` ne sont plus nécessaires. Comme ils ne sont plus nécessaires, vous pouvez supprimer le répertoire `/tools` si vous le voulez ou l'archiver dans un fichier tar et le conserver pour construire un autre système.



### Note

Supprimer `/tools` effacera aussi les copies temporaires de Tcl, Expect et DejaGnu, qui ont été utilisé pour lancer les tests de l'ensemble des outils. Si vous avez besoin de ces programmes plus tard, vous devrez les recompiler et les ré-installer. Le livre BLFS a les bonnes instructions pour le faire.

# Chapitre 7. Initialiser les scripts de démarrage du système

## 7.1. Introduction

Ce chapitre montre comment installer et configurer les scripts de démarrage LFS. La plupart de ces scripts fonctionne sans modification mais quelques-uns nécessitent des fichiers de configuration supplémentaires car ils utilisent 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://www.linuxfromscratch.org/hints/downloads/files/bsd-init.txt>. Rechercher dans les listes de diffusion LFS vous offrira aussi d'autres choix.

Si vous utilisez un autre style de scripts de démarrage, passez ce chapitre et allez directement sur le Chapitre 8.

## 7.2. LFS-Bootscripts-3.2.1

Le paquet LFS-Bootscripts contient un ensemble de scripts de démarrage pour démarrer/arrêter le système LFS.

**Temps de construction estimé :** 0,1 SBU

**Espace disque requis :** 0,3 Mo

**Dépendances de l'installation :** Bash et Coreutils

### 7.2.1. Installation de LFS-Bootscripts

Installez le paquet :

```
make install
```

### 7.2.2. Contenu de LFS-bootscripts

**Scripts installés:** checkfs, cleanfs, console, functions, halt, hotplug, ifdown, ifup, localnet, mountfs, mountkernfs, network, rc, reboot, sendsignals, setclock, static, swap, sysklogd, template et udev

#### Descriptions courtes

<b>checkfs</b>	Vérifie l'intégrité des systèmes de fichiers avant de les monter (avec l'exception des systèmes de fichiers journalisés ou réseau)
<b>cleanfs</b>	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> qui pourraient être présents
<b>console</b>	Charge la bonne table de correspondance du clavier ; il initialise aussi la police de l'écran
<b>functions</b>	Contient des fonctions communes, telles que la vérification d'erreurs et de statuts, utilisées par les différents scripts.
<b>halt</b>	Arrête le système
<b>hotplug</b>	Charge des modules pour les périphériques système
<b>ifdown</b>	Assiste le script <code>network</code> pour l'arrêt des périphériques réseaux
<b>ifup</b>	Assiste le script <code>network</code> pour le démarrage des périphériques réseaux
<b>localnet</b>	Configure le nom d'hôte du système et le périphérique de boucle locale
<b>mountfs</b>	Monte tous les systèmes de fichiers, sauf ceux marqués <i>noauto</i> et les systèmes réseaux
<b>mountkernfs</b>	Monte les systèmes de fichiers virtuels fournies par le noyau, tels que <code>proc</code>
<b>network</b>	Configure les interfaces réseaux, telles que les cartes réseaux, et configure la passerelle par défaut (lorsque c'est applicable)
<b>rc</b>	Le 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



<b>reboot</b>	Redémarre le système
<b>sendsignals</b>	S'assure que chaque processus est terminé avant que le système redémarre ou ne s'arrête
<b>setclock</b>	Réinitialise l'horloge noyau avec l'heure locale au cas où l'horloge matérielle n'est pas en temps UTC
<b>static</b>	Fournit les fonctionnalités nécessaires à l'affectation d'une adresse statique IP (Internet Protocol) vers une interface réseau
<b>swap</b>	Active et désactive les fichiers swap et les partitions
<b>syslogd</b>	Lance et arrête les démons des journaux système et noyau
<b>template</b>	Un modèle pour créer des scripts de démarrage personnalisés pour d'autres démons
<b>udev</b>	Prépare le répertoire /dev et lance Udev.

## 7.3. Fonctionnement des 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 LFS. LFS a sa propre façon de le faire mais il respecte généralement les standards établis.

SysVinit (qui sera nommé par la suite « init ») fonctionne en utilisant un schéma de niveaux d'exécution. Ils sont au nombre de sept (numérotés de 0 à 6). En fait, il en existe plus mais ils sont 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_exécution]`, où `[niveau_exécution]` est le niveau d'exécution cible. Par exemple, pour redémarrer l'ordinateur, un utilisateur devra lancer la commande `init 6` qui est un alias de la commande `reboot`. De la même façon, `init 0` est un alias de 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` argument.

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.

## 7.4. Gestion des périphériques et modules sur un système LFS

Dans Chapitre 6, 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), quel que soit 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.

### 7.4.1. Historique

En février 2000, un nouveau système de fichiers appelé `devfs` a été intégré au noyau 2.3.46 et rendu disponible pour la série 2.4 des noyaux stables. Bien qu'il soit présent dans 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 gérait la détection, la création et le nommage des périphériques. Ce dernier problème, le nommage des périphériques, était peut-être le plus critique. Il est généralement accepté que s'il est possible de configurer les noms des périphériques, alors la politique de nommage des périphériques revient à l'administrateur du système, et du coup n'est pas imposée par un développeur 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 travail 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.

### 7.4.2. Implémentation d'Udev

Le système de fichiers `sysfs` a été brièvement mentionné ci-dessus. On pourrait se demander comment `sysfs` connaît les périphériques présents sur un système et quels numéros de périphériques devraient être utilisés. Les pilotes qui ont été compilés directement dans le noyau enregistrent 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 fichiers `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.

Le script de démarrage **S10udev** fait attention à créer les nœuds périphériques au lancement de Linux. Ce script commence en enregistrant `/sbin/udevsend` comme gestionnaire d'événements de montage à chaud. Ces événements (discutés plus bas) ne sont généralement pas générés lors de cette étape mais **udev** est enregistré juste au cas où cela se passerait quand même. Le programme **udevstart** parcourt le système de fichiers `/sys` et crée les périphériques sous `/dev` correspondant à ces descriptions. Par exemple, `/sys/class/tty/vcs/dev` contient la chaîne « 7:0 ». Cette chaîne est utilisée par **udevstart** pour

créer `/dev/vcs` avec le nombre majeur 7 et le nombre mineur 0. Les noms et droits des nœuds créés sous le répertoire `/dev` sont configurés suivant les règles spécifiées dans les fichiers du répertoire `/etc/udev/rules.d/`. Ils sont numérotés d'une façon similaire au packaging LFS-Bootscripts. Si **udev** ne peut pas trouver une règle pour le périphérique en cours de création, celui-ci aura les droits par défaut (660) et son propriétaire sera `root:root`.

Une fois l'étape ci-dessus terminée, tous les périphériques qui étaient déjà présents et ont des pilotes intégrés au noyau seront disponibles pour utilisation. Ceci nous amène aux périphériques qui ont des pilotes sous forme de modules.

Plus tôt, nous avons mentionné le concept d'un « gestionnaire d'événements de montage à chaud ». Quand la connexion d'un nouveau périphérique est détectée par le noyau, le noyau générera un événement de montage à chaud et regardera dans le fichier `/proc/sys/kernel/hotplug` pour trouver le programme en espace utilisateur qui gère la connexion du périphérique. Le script de démarrage **udev** a enregistré **udevsend** comme gestionnaire. Quand ces événements sont générés, le noyau indiquera à **udev** de vérifier le système de fichiers `/sys` pour des informations sur le nouveau périphérique et pour créer son entrée `/dev`.

Ceci nous amène au problème d'**udev**, mais aussi avec `devfs`. Il est habituellement référencé comme le « problème de l'oeil;uf et de la poule ». La plupart des distributions Linux gère le chargement des modules via des entrées dans `/etc/modules.conf`. L'accès à un nœud périphérique implique le chargement du module du noyau. Avec **udev**, cette méthode ne fonctionnera pas car le nœud périphérique n'existe pas tant que le module n'est pas chargé. Pour résoudre ceci, le script de démarrage **S05modules** a été ajouté au paquet `lfs-bootscripts`, avec le fichier `/etc/sysconfig/modules`. En ajoutant les noms de modules au fichier `modules`, ces modules seront chargés lorsque l'ordinateur démarrera. Ceci permet à **udev** de détecter les périphériques et de créer les nœuds périphériques appropriés.

Notez que sur les machines lentes ou pour les pilotes qui créent un grand nombre de nœuds périphériques, le processus de création des périphériques pourrait prendre quelques secondes pour se terminer. Ceci signifie que certains nœuds périphériques pourraient ne pas être accessibles immédiatement.

### 7.4.3. Gestion des périphériques dynamiques/montables à chaud

Lorsque vous connectez un périphérique, comme un lecteur MP3 USB, le noyau reconnaît que le périphérique est maintenant connecté et génère un événement de montage à chaud. Si le pilote a déjà été chargé (soit parce qu'il était compilé dans le noyau soit parce qu'il a été chargé via le script de démarrage **S05modules**), **udev** sera appelé pour créer le(s) nœud(s) périphérique correspondant suivant les données de `sysfs` disponibles dans `/sys`.

Si le pilote du périphérique tout juste connecté est disponible comme module mais actuellement non chargé, le packaging Hotplug chargera les modules appropriés et rendra ce périphérique disponible en créant le(s) nœud(s) périphérique(s) qui le concernent.

### 7.4.4. Problèmes avec la création des périphériques

Il existe quelques problèmes connus pour la création automatique des nœuds périphériques :

1) Un pilote du noyau pourrait ne pas exporter ses données dans `sysfs`.

Ceci est plus fréquent avec les pilotes de tierces parties, à l'extérieur du noyau. Udev ne sera pas capable de créer automatiquement les nœuds périphériques pour de tels pilotes. Utilisez le fichier de configuration `/etc/sysconfig/createfiles` pour créer manuellement les périphériques. Consultez le fichier `devices.txt` dans la documentation du noyau ou la documentation pour trouver les bons numéros majeurs et mineurs pour ce périphérique.

2) Un périphérique non matériel est requis. Ceci est plus fréquent avec le module de compatibilité OSS

(acronyme de Open Sound System) du projet ALSA (Advanced Linux Sound Architecture). Ces types de périphériques peuvent être gérés de deux façons différentes :

- Ajouter les noms des modules à `/etc/sysconfig/modules`

- Utiliser une ligne « install » dans `/etc/modprobe.conf`. Ceci indique à la commande **modprobe** « au chargement du module, de charger aussi cet autre module, au même moment. » Par exemple :

```
install snd-pcm modprobe -i snd-pcm ; modprobe \  
snd-pcm-oss ; true
```

Ceci fera que le système charge à la fois les modules *snd-pcm* et *snd-pcm-oss* quand toute requête est faite pour charger le pilote *snd-pcm*.

## 7.4.5. Lecture utile

Des documentations supplémentaires sont disponibles sur les sites suivants :

- A Userspace Implementation of `devfs` [http://www.kroah.com/linux/talks/ols\\_2003\\_udev\\_paper/Reprint-Kroah-Hartman-OLS2003.pdf](http://www.kroah.com/linux/talks/ols_2003_udev_paper/Reprint-Kroah-Hartman-OLS2003.pdf) (NdT : Une implémentation en espace utilisateur de `devfs`)
- FAQ `udev` <http://www.kernel.org/pub/linux/utils/kernel/hotplug/udev-FAQ>
- The Linux Kernel Driver Model [http://public.planetmirror.com/pub/lca/2003/proceedings/papers/Patrick\\_M](http://public.planetmirror.com/pub/lca/2003/proceedings/papers/Patrick_M) (NdT : Le modèle du pilote pour le noyau Linux)

## 7.5. Configurer le script `setclock`

Le script `setclock` lit le temps sur l'horloge matérielle, aussi connu sous le nom d'horloge BIOS or 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 dans le fuseau horaire MST, aussi connu en tant que GMT -0700, ajoutez sept heures à l'heure locale.

Modifiez la valeur de la variable UTC ci-dessous par une valeur 0 (zéro) si l'horloge matérielle n'est *pas* configurée en temps UTC.

Créez un nouveau fichier `/etc/sysconfig/clock` en lançant ce qui suit :

```
cat > /etc/sysconfig/clock << "EOF"
# Début /etc/sysconfig/clock

UTC=1

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

Une bonne astuce expliquant comment gérer l'horloge sur LFS est disponible sur <http://www.linuxfromscratch.org/hints/downloads/files/time.txt>. Il explique certains concepts comme les fuseaux horaires, UTC et la variable d'environnement TZ.



## 7.6. Configurer la console Linux

Cette section discute de la configuration du script de démarrage **console**, 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 **console** lit le fichier `/etc/sysconfig/console` pour des informations de configuration. Il décide du plan de codage et de la police de la console à utiliser. Différents guides pratiques spécifiques aux langues peuvent aussi être d'une grande aide (voir <http://www.tldp.org/HOWTO/HOWTO-index/other-lang.html>). Un fichier `/etc/sysconfig/console` déjà rempli avec des paramètres déjà connus pour différents pays a été installé avec la paquet LFS-Bootscripts, pour que la section adéquate puisse être décommentée si le pays est supporté. Si vous avez toujours des doutes, jetez un œil dans le répertoire `/usr/share/kbd` pour des plans de codage valides et des polices pour écran. Lisez les pages man de `loadkeys(1)` et `setfont(8)` pour déterminer les bons arguments pour ces programmes. Une fois décidé, créez le fichier de configuration avec les commandes suivantes :

```
cat >/etc/sysconfig/console <<"EOF"
KEYMAP="[arguments pour loadkeys]"
FONT="[arguments pour setfont]"
EOF
```

Par exemple, pour les utilisateurs espagnols qui souhaitent aussi utiliser le caractère Euro (accessible avec la combinaison de touches AltGr+E), les paramètres suivants sont corrects :

```
cat >/etc/sysconfig/console <<"EOF"
KEYMAP="es euro2"
FONT="lat9-16 -u iso01"
EOF
```



### Note

La ligne FONT ci-dessus est correct seulement pour l'ensemble de caractères ISO 8859-15. Si vous utilisez ISO 8859-1 et, du coup, un symbole de livre à la place de l'Euro, la ligne FONT correcte devrait être :

```
FONT="lat1-16"
```

Si la variable `KEYMAP` ou `FONT` n'est pas configurée, le script de démarrage **console** n'exécutera pas le programme correspondant.

Dans certains plans de codage, les touches Backspace et Delete envoient des caractères différents de ceux du plan de codage par défaut intégré au noyau. Ceci est une source de confusion pour certaines applications. Par exemple, Emacs affiche son aide (au lieu de supprimer le caractère situé avant le curseur) lors d'un appui sur Backspace. Pour vérifier si le plan de codage utilisé est pris en compte (ceci ne fonctionne qu'avec les plans de codage i386) :

```
zgrep '\Wl4\W' [ /path/to/your/keymap ]
```

Si le code de touche 14 est Backspace au lieu de Delete, créez le code supplémentaire suivant au plan de codage pour corriger ce problème :

```
mkdir -pv /etc/kbd && cat > /etc/kbd/bs-sends-del <<"EOF"
        keycode 14 = Delete Delete Delete Delete
    alt keycode 14 = Meta_Delete
altgr alt keycode 14 = Meta_Delete
        keycode 111 = Remove
altgr control keycode 111 = Boot
    control alt keycode 111 = Boot
altgr control alt keycode 111 = Boot
EOF
```

Indiquez au script **console** de charger ce code après le plan de codage principal :

```
cat >>/etc/sysconfig/console <<"EOF"
KEYMAP_CORRECTIONS="/etc/kbd/bs-sends-del "
EOF
```

Pour compiler le plan de codage directement dans le noyau au lieu de le configurer à chaque fois avec le script de démarrage **console**, suivez les instructions données dans Section 8.3, « Linux-2.6.11.12. ». Le faire vous assure que le plan de codage fonctionnera toujours comme attendu, même si vous démarrez en mode maintenance (en passant *init=/bin/sh* au noyau), parce que le script de démarrage **console** ne sera pas lancé dans cette situation. De plus, le noyau ne configurera plus la police de l'écran automatiquement. Ceci ne devrait pas poser beaucoup de problèmes car les caractères ASCII seront gérés correctement et il est improbable qu'un utilisateur aura besoin de caractères non ASCII en mode maintenance.

Comme le noyau configurera le plan de codage, il est possible d'omettre la variable `KEYMAP` à partir du fichier de configuration `/etc/sysconfig/console`. Il peut aussi être laissé à sa place, si désiré, sans conséquence. Le garder pourrait être bénéfique si vous utilisez différents noyaux et qu'il est difficile d'être sûr que le plan de codage est bien compilé dans chacun d'entre eux.

## 7.7. Configurer le script `sysklogd`

Le script `sysklogd` invoque le programme `syslogd` avec l'option `-m`. Cette option désactive la marque périodique que `syslogd` écrit sur les journaux système toutes les vingt minutes par défaut. Si vous voulez l'activer, éditez le script `sysklogd` et faites les changements adéquats. Voir la page de manuel `man syslogd` pour plus d'informations.

## 7.8. 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` 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 la section *Readline Init File* dans **info bash**. **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. Notez 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 /etc/inputrc
# Modifié par Chris Lynn <roryo@roryo.dynup.net>

# Autorise l'invite de commande de passer à la ligne suivant
set horizontal-scroll-mode Off

# Active la saisie 8bit
set meta-flag On
set input-meta On

# Désactive la suppression du 8ème bit
set convert-meta Off

# Conserve le 8ème bit pour l'affichage
set output-meta On

# aucun, visible ou audible
set bell-style none

# Tout ce qui suit fait correspondre la séquence d'échappement
# de la valeur contenue à l'intérieur du premier argument aux
# fonctions spécifiques 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 les xterm
"\eOH": beginning-of-line
"\eOF": end-of-line
```

```
# pour Konsole
"\e[H": beginning-of-line
"\e[F": end-of-line

# Fin /etc/inputrc
EOF
```

## 7.9. Les 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 `[invite]$ /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*.

Les fichiers `/etc/profile` et `~/.bash_profile` sont lus quand le shell est appelé en tant que shell interactif de connexion.

Le fichier `/etc/profile` de base, ci-dessous, configure quelques variables d'environnement nécessaire au support des langues natives. Les configurer proprement résulte en ce qui suit :

- La sortie des programmes traduite dans le langage natif
- Correction de la classification des caractères en lettres, chiffres et autres classes. Ceci est nécessaire pour que **bash** accepte proprement les caractères non ASCII dans les lignes de commandes pour les locales autres qu'anglais
- L'ordre de tri alphabétique correct pour le pays
- Taille de papier par défaut appropriée
- Bon formatage des valeurs monétaires, de l'heure et des dates

Ce script configure aussi la variable d'environnement `INPUTRC` qui fait que Bash et Readline utilisent le fichier `/etc/inputrc` créé précédemment.

Remplacez `[LL]` ci-dessous avec le code à deux lettres de la langue désirée (par exemple, « en ») et `[CC]` avec le code à deux lettres du pays approprié (par exemple, « GB »). `[charmap]` devra être remplacé avec le charmap canonique de votre locale.

La liste de toutes les locales supportées par Glibc peut être obtenue en exécutant la commande suivante :

```
locale -a
```

Les locales peuvent avoir plusieurs synonymes. Par exemple, « ISO-8859-1 » est aussi appelée « iso8859-1 » et « iso88591 ». Quelques applications ne peuvent pas gérer les différents synonymes correctement, donc il est plus sûr de choisir le nom canonique pour une locale particulière. Pour déterminer le nom canonique, lancez la commande suivante, où `[nom locale]` est l'affichage donnée par **locale -a** pour votre locale préférée (« en\_GB.iso88591 » dans notre exemple).

```
LC_ALL=[nom locale] locale charmap
```

Pour la locale « en\_GB.iso88591 », la commande ci-dessus affichera :

```
ISO-8859-1
```

Ceci résulte en un paramétrage final de locale avec « en\_GB.ISO-8859-1 ». Il est important que la locale trouvée utilisant l'heuristique ci-dessus soit testée avant d'être ajoutée aux fichiers de démarrage de Bash :

```
LC_ALL=[locale name] locale country
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
```

Les commandes ci-dessus devraient afficher les noms du pays et de la langue, le codage des caractères utilisé par la locale, la monnaie et le préfixe à composer avant de saisir le numéro de téléphone. Si une des commandes ci-dessus échoue avec un message similaire à un de ceux montrés ci-dessous, cela signifie que votre locale n'a pas été installée dans le chapitre 6 ou qu'elle n'est pas supportée par l'installation par défaut de Glibc.

```
locale: Cannot set LC_* to default locale: No such file or directory
```

Si cela arrive, vous pouvez soit installer la locale désirée en utilisant la commande **localedef** soit considérer l'utilisation d'une locale différente. Les instructions suivantes supposent qu'il n'y a pas eu de tels messages de Glibc.

Certains paquets en dehors de LFS pourraient aussi ne pas avoir de support pour la locale que vous avez choisi. Un exemple est la bibliothèque X (qui fait partie du système X Window), qui affiche le message d'erreur suivant :

```
Warning: locale not supported by Xlib, locale set to C
```

Quelque fois, il est possible de corriger ceci en supprimant la partie charmap de la spécification de la locale si cela ne modifie pas le "character map" que Glibc associe avec la locale (ceci peut être vérifié en exécutant la commande **locale charmap** dans les deux locales). Par exemple, vous pouvez changer "de\_DE.ISO-8859-15@euro" en "de\_DE@euro" pour obtenir que cette locale soit reconnue par Xlib.

D'autres paquets peuvent aussi mal fonctionner (mais pourraient ne pas nécessairement afficher de messages d'erreurs) si le nom de la locale ne correspond pas à leur attente. Dans de tels cas, vous pouvez obtenir des informations utiles en cherchant comme les autres distributions Linux supportent votre locale.

Une fois que les bons paramètres de locale ont été déterminés, créez le fichier `/etc/profile` :

```
cat > /etc/profile << "EOF"
# Début /etc/profile

export LANG=[ll]_[CC].[charmap]
export INPUTRC=/etc/inputrc

# Fin /etc/profile
EOF
```



### Note

Les locales « C » (par défaut) et « en\_US » (celle recommandée pour les utilisateurs de langue anglaise vivant aux États-Unis) sont différentes.

Initialiser le plan de codage du clavier, la police de la console et les variables d'environnement relatives à la locale sont les seules étapes d'internationalisation nécessaires pour supporter les locales qui utilisent habituellement les codages à un seul octet et la direction d'écriture de la gauche vers la droite. Les cas plus complexes (incluant les locales basées sur UTF-8) nécessitent des étapes et des correctifs supplémentaires parce qu'un grand nombre d'applications ont tendance à ne pas fonctionner correctement dans de telles conditions. Ces étapes et correctifs ne sont pas inclus dans le livre LFS et de telles locales ne sont pas supportées par LFS.



## 7.10. Configurer le script localnet

Une partie du boulot de ce script 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=[lfs]" > /etc/sysconfig/network
```

`[lfs]` doit être remplacé par le nom de l'ordinateur. Ne saisissez pas le FQDN (Fully Qualified Domain Name, nom de domaine pleinement qualifié) ici. Cette information sera rentrée dans le fichier `/etc/hosts` dans la prochaine section.

## 7.11. Créer 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 la suivante :

```
<adresse IP> mon-hôte.mon-domaine.org aliases
```

À moins que votre ordinateur ne 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'a 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 :

Classes	Réseaux
A	10.0.0.0
B	172.16.0.0 à 172.31.0.255
C	192.168.0.0 à 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` (ceci n'est pas recommandé car il s'agit d'un nom de domaine valide enregistré et que cela pourrait causer des problèmes à votre serveur).

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` en lançant la commande :

```
cat > /etc/hosts << "EOF"
# Début de /etc/hosts (version avec carte réseau)

127.0.0.1 localhost
[192.168.1.1] [<nom d'hôte>.exemple.org] [nom d'hôte]

# Fin de /etc/hosts (version avec carte réseau)
EOF
```

Les valeurs `[192.168.1.1]` et `[<nom d'hôte>.exemple.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 la commande :

```
cat > /etc/hosts << "EOF"
# Début /etc/hosts (version sans carte réseau)

127.0.0.1 [<nom d'hôte>.exemple.org] [nom d'hôte] localhost

# Fin /etc/hosts (version sans carte réseau)
EOF
```

## 7.12. Configurer le script network

Cette section s'applique seulement si une carte réseau doit être configurée.

Si une carte réseau ne sera pas utilisée, il n'y a aucun besoin de créer des fichiers de configuration relatifs aux cartes réseau. Si c'est le cas, supprimez les liens symboliques `network` de tous les répertoires des niveaux d'exécution (`/etc/rc.d/rc*.d`).

### 7.12.1. Créer des fichiers de configuration des interfaces réseau

Les interfaces activées et désactivées par le script `network` dépendent des fichiers et des répertoires compris dans le répertoire `/etc/sysconfig/network-devices`. Ce répertoire doit contenir un sous-répertoire pour chaque interface à configurer, comme `ifconfig.xyz`, avec « xyz » le nom de l'interface réseau. Dans ce répertoire se trouvent des fichiers définissant les attributs de cette interface, comme le(s) adresse(s) IP, masque de sous-réseau et ainsi de suite.

La commande suivante crée un fichier `ipv4` d'exemple pour le périphérique `eth0` :

Les nouveaux fichiers sont créés dans ce répertoire. La commande suivante crée un fichier d'exemple `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
```

Les valeurs de ces variables doivent être modifiées dans chaque fichier pour correspondre à la bonne configuration. Si la variable `ONBOOT` est configurée à « yes », le script `network` configurera la carte réseau (NIC) pendant le démarrage du système. S'il est configuré avec toute autre valeur que « yes », l'interface réseau sera ignorée par le script `network` et non montée.

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

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

La variable `PREFIX` a besoin de contenir le nombre de bits utilisé dans le sous-réseau. Chaque octet dans une adresse IP est sur huit bits. Si le masque réseau du sous-réseau est `255.255.255.0`, alors il est en train d'utiliser les trois premiers octets (24 bits) pour spécifier le numéro réseau number. Si le masque réseau est `255.255.255.240`, il utiliserait les 128 premiers bits. Les préfixes plus longs que 24 bits sont habituellement utilisés par les fournisseurs d'accès à Internet DSL et câble. Dans cet exemple (`PREFIX=24`), le masque réseau est `255.255.255.0`. Ajustez la variable `PREFIX` en concordance avec votre sous-réseau spécifique.

### 7.12.2. Créer le fichier `/etc/resolv.conf`

Si le système a besoin d'être connecté à Internet, il aura besoin de la résolution de noms proposée par le DNS (Domain Name Service) pour résoudre les noms de domaines Internet, et vice-versa. Ceci se fait en

plaçant les adresses IP du serveur DNS, disponibles auprès du FAI ou de l'administrateur système, dans `/etc/resolv.conf`. Créez le fichier en lançant ce qui suit :

```
cat > /etc/resolv.conf << "EOF"
# Début /etc/resolv.conf

domain {[Votre nom de domaine]}
nameserver [adresse IP de votre DNS principal]
nameserver [adresse IP de votre DNS secondaire]

# Fin /etc/resolv.conf
EOF
```

Remplacez *[adresse IP du DNS]* avec l'adresse IP du DNS le plus approprié pour la configuration. Il y aura souvent plus d'une entrée (les conseils recommandent des serveurs DNS disposant de capacité de prise en charge. Si vous avez seulement besoin ou si vous voulez uniquement le serveur DNS, supprimez la seconde ligne *serveur de noms* à partir du fichier. L'adresse IP pourrait aussi être un routeur sur le réseau local.

## Chapitre 8. Rendre le système LFS amorçable

### 8.1. Introduction

Il est temps de rendre amorçable le système LFS. Ce chapitre traite de la création d'un fichier `fstab`, de la construction d'un noyau pour le nouveau système LFS et de l'installation du chargeur de démarrage Grub afin que le système LFS puisse être sélectionné au démarrage.

## 8.2. Créer le fichier `/etc/fstab`

Le fichier `/etc/fstab` est utilisé par quelques programmes pour déterminer les 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 /etc/fstab

# système      point          type  options          dump  order pour fsck
# de fichiers  de montage

/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
# Fin /etc/fstab
EOF
```

Bien sûr, remplacez `[xxx]`, `[yyy]` et `[fff]` avec les valeurs appropriées pour votre système -- par exemple `hda2`, `hda5` et `reiserfs`. Pour tous les détails sur les six champs de cette table, voir **man 5 fstab**.

Lors de l'utilisation d'un système de fichiers journalisé, le `1 1` à la fin de la ligne doit être remplacé par `0 0` car une telle partition ne doit pas être « dumpée » ou vérifiée.

Le point de montage `/dev/shm` pour `tmpfs` est inclus 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 utilisent la mémoire partagée POSIX. Donc, vous pouvez considérer le point de montage `/dev/shm` optionnel. Pour plus d'informations, voir `Documentation/filesystems/tmpfs.txt` dans le répertoire des sources du noyau.

Il existe d'autres lignes qui pourraient être ajoutées à votre fichier `/etc/fstab`. Un exemple entre autres est la ligne pour les périphériques USB :

```
usbfs          /proc/bus/usb usbfs  devgid=14,devmode=0660 0 0
```

Cette option fonctionnera seulement si « Support for Host-side USB » et « USB device filesystem » sont configurés dans le noyau, et non pas en tant que modules. Si « Support for Host-side USB » est compilé comme module, alors `usbcore` doit être indiqué dans `/etc/sysconfig/modules`.

## 8.3. Linux-2.6.11.12

Le paquet Linux contient le noyau Linux.

**Temps de construction estimé :** 4,20 SBU

**Espace disque requis :** 181 Mo

**Dépendances de l'installation :** Bash, Binutils, Coreutils, Findutils, GCC, Glibc, Grep, Gzip, Make, Modutils, Perl et Sed

### 8.3.1. Installation du noyau

Construire le noyau implique un certain nombre d'étapes : configuration, compilation et installation. Lisez le fichier README contenu dans les sources du noyau pour d'autres méthodes.

Préparez la compilation en lançant la commande suivante :

```
make mrproper
```

Ceci nous assure que le répertoire du noyau est complètement nettoyé. L'équipe du noyau recommande que cette commande soit lancée avant chaque compilation du noyau. Vous ne devez pas penser que le répertoire des sources est propre juste après avoir été déballé.

S'il a été décidé dans Section 7.6, « Configurer la console Linux, » de compiler le plan de codage dans le noyau, lancez la commande ci-dessous :

```
loadkeys -m /usr/share/kbd/keymaps/[path to keymap] > \
drivers/char/defkeymap.c
```

Par exemple, pour utiliser un clavier hollandais, utilisez `/usr/share/kbd/keymaps/i386/qwerty/nl.map.gz`.

Configurez le noyau via une interface par menu. BLFS a quelques informations concernant les besoins particuliers du noyau en terme de configuration pour les paquetages en dehors de LFS sur <http://www.linuxfromscratch.org/blfs/view/svn/longindex.html#kernel-config-index> :

```
make menuconfig
```

Sinon, **make oldconfig** peut être plus approprié dans certaines situations. Voir le fichier README pour plus d'informations.

Si désiré, passez la configuration du noyau en copiant le fichier de configuration, `.config`, à partir du système hôte (en supposant qu'il est disponible) dans le répertoire `linux-2.6.11.12` tout juste déballé. Néanmoins, nous ne recommandons pas cette option. Il est souvent mieux d'explorer tous les menus de configuration et de créer la configuration du noyau à partir de rien.



#### Note

NPTL requiert que le noyau soit compilé avec GCC 3.x ou ultérieur, dans ce cas 3.4.3. Il n'est pas recommandé de compiler le noyau avec GCC-2.95.x car ceci est la cause d'échec dans la suite de tests de Glibc. Normalement, ceci ne devrait pas être mentionné car LFS ne construit pas GCC-2.95.x. Malheureusement, la documentation du noyau est obsolète et indique toujours que GCC-2.95.3 est le compilateur recommandé.

Compilez l'image du noyau et les modules :

```
make
```

Si vous utilisez des modules avec le noyau, un fichier `/etc/modprobe.conf` pourrait être nécessaire. Les informations concernant les modules et la configuration du noyau sont situées dans la documentation du noyau disponible dans le répertoire `linux-2.6.11.12/Documentation`. De plus, la page `man modprobe.conf(5)` pourrait aussi avoir de l'intérêt.

Faites attention lors de la lecture d'autres documentations relatives aux modules du noyau parce qu'elles s'appliquent généralement seulement aux noyaux 2.4.x. D'après nos connaissances, les problèmes spécifiques de configuration du noyau pour Hotplug (le montage à chaud) et Udev se sont pas documentés. Le problème est que Udev créera un nœud de périphérique seulement si Hotplug ou un script écrit par l'utilisateur insère le module correspondant dans le noyau et tous les modules ne sont pas détectables par Hotplug. Notez que les instructions comme celle ci-dessous, compris dans le fichier `/etc/modprobe.conf`, ne fonctionnent pas avec Udev :

```
alias char-major-XXX un-module
```

À cause des complications avec Hotplug, Udev et les modules, nous recommandons fortement de commencer avec une configuration complètement non modulaire du noyau, spécialement si c'est la première fois que vous utilisez Udev.

Installez les modules si la configuration du noyau les utilise :

```
make modules_install
```

Une fois la compilation du noyau terminée, des étapes supplémentaires sont requises pour terminer l'installation. Certains fichiers ont besoin d'être copiés dans le répertoire `/boot`.

Le chemin vers l'image du noyau pourrait varier suivant la plateforme d'utilisation. La commande suivante suppose qu'elle se trouve sur une architecture x86 :

```
cp -v arch/i386/boot/bzImage /boot/lfskernel-2.6.11.12
```

`System.map` est un fichier de symboles pour le noyau. Il cartographie les points d'entrées de chaque fonction dans l'API du noyau, ainsi que les adresses des structures de données du noyau pour le noyau en cours d'exécution. Lancez la commande suivante pour installer le fichier carte :

```
cp -v System.map /boot/System.map-2.6.11.12
```

Le fichier de configuration du noyau `.config` produit à l'étape **make menuconfig** ci-dessus contient toutes les sélections de configuration pour le noyau tout juste compilé. Conserver ce fichier est une bonne idée pour pouvoir s'y référer plus tard :

```
cp -v .config /boot/config-2.6.11.12
```



Il est important de noter que les fichiers du répertoire des sources du noyau n'appartiennent pas à *root*. À chaque fois qu'un paquet est déballé par l'utilisateur *root* (comme nous l'avons fait à l'intérieur du chroot), les fichiers conservent les identifiants utilisateur et groupe de celui qui a préparé le paquet. Ceci n'est habituellement pas un problème pour tout autre paquet installé car le répertoire des sources est supprimé après installation. Néanmoins, le répertoire des sources de Linux est souvent conservé assez longtemps. À cause de cela, il existe une chance pour que l'identifiant utilisateur soit affecté à quelqu'un sur la machine. Cette personne aurait alors les droits d'écriture sur les sources du noyau.

Si le répertoire des sources du noyau est conservé, lancez **chown -R 0:0** sur le répertoire `linux-2.6.11.12` pour vous assurer que tous les fichiers sont bien possédés par l'utilisateur *root*.



### Avertissement

Certaines documentations du noyau recommandent de créer un lien symbolique à partir de `/usr/src/linux` vers le répertoire des sources du noyau. Ceci est spécifique aux noyaux antérieurs à la série 2.6 et *ne doit pas* être réalisé sur un système LFS car il peut poser des problèmes pour les paquetages que vous souhaitez construire une fois que votre système LFS de base est complet.

De plus, les en-têtes compris dans le répertoire système `include` devraient *toujours* être ceux avec lesquels Glibc a été compilé et, du coup, ne devraient *jamais* être remplacés par les en-têtes du noyau.

## 8.3.2. Contenu de Linux

**Fichiers installés:** `config-2.6.11.12`, `lfskernel-2.6.11.12`, et `System.map-2.6.11.12`

### Descriptions courtes

<code>config-2.6.11.12</code>	Contient toutes les sélections de la configuration pour le noyau
<code>noyau</code>	Le moteur du système Linux. Au démarrage de l'ordinateur, le noyau est la première partie du système d'exploitation à être chargée. Il détecte et initialise tous les composants matériels de l'ordinateur, puis rend disponibles ces composants par un ensemble de fichiers pour les logiciels qui en ont besoin, et transforme un CPU unique en une machine multitâches capable d'exécuter des bouts de programmes quasiment au même moment.
<code>System.map-2.6.11.12</code>	Une liste d'adresses et de symboles ; il fait correspondre les points d'entrées et les adresses de toutes les fonctions et structures de données dans le noyau

## 8.4. Rendre le système LFS amorçable

Votre nouveau système LFS est pratiquement fini. Une des dernières choses à faire est de vous assurer que le système peut démarrer proprement. Les instructions ci-dessous s'appliquent seulement aux ordinateurs de l'architecture IA-32, c'est-à-dire les PC standards. Des informations sur le « chargement au démarrage » pour les autres architectures devraient être disponibles aux emplacements habituels des ressources pour ces architectures.

Le chargement au démarrage est un domaine complexe. Tout d'abord, quelques mots de mise en garde sont nécessaires. Vous devez vraiment connaître le chargeur actuel et tout autre système d'exploitation présent sur le disque dur amorçable. Assurez-vous d'avoir une disquette de démarrage de façon à pouvoir réparer l'ordinateur si, par malheur, celui-ci devenait inutilisable (non amorçable).

Plus tôt, nous avons compilé et installé le chargeur de démarrage Grub pour cette étape. La procédure implique l'écriture de quelques fichiers spéciaux de Grub en des endroits spécifiques sur le disque dur. Nous recommandons fortement la création d'une disquette de démarrage Grub comme sauvegarde. Insérez une disquette de démarrage vierge et lancez les commandes suivantes :

```
dd if=/boot/grub/stage1 of=/dev/fd0 bs=512 count=1
dd if=/boot/grub/stage2 of=/dev/fd0 bs=512 seek=1
```

Enlevez la disquette et rangez-la dans un endroit sûr. Maintenant, lancez le shell **grub** :

```
grub
```

Grub utilise sa propre structure de nommage des disques et partitions, de la forme  $(hdn,m)$ , où  $n$  est le numéro du disque dur et  $m$  le numéro de la partition, tout deux commençant à zéro. Par exemple, la partition `hda1` est  $(hd0,0)$  pour Grub alors que `hdb3` est  $(hd1,2)$ . Contrairement à Linux, Grub ne considère pas les lecteurs de CDROMs comme des disques durs. Par exemple, si un CD se trouve sur `hdc` et un second disque dur sur `hdc`, ce dernier disque sera malgré tout  $(hd1)$ .

En utilisant les informations ci-dessus, déterminez la désignation appropriée pour votre partition root (ou votre partition de démarrage si celle que vous utilisez est séparée). Pour l'exemple suivant, il est supposé que votre partition root (ou votre partition séparée) est `hda4`.

Indiquez à Grub où chercher ses fichiers `stage{1,2}`. La touche tabulation est utilisable partout pour que Grub vous affiche les alternatives :

```
root (hd0,3)
```



### Avertissement

La commande suivante écrasera votre chargeur de démarrage actuel. Ne lancez pas cette commande si ce n'est pas désiré, par exemple, lors de l'utilisation d'un autre gestionnaire de démarrage pour gérer votre MBR (Master Boot Record). Dans ce cas, il serait probablement plus sensé d'installer Grub dans le « secteur de boot » de la partition LFS, auquel cas la prochaine commande deviendrait : **setup (hd0,3)**.

Ensuite, indiquez à Grub de s'installer dans le MBR de `hda` :

```
setup (hd0)
```

Si tout va bien, Grub indiquera avoir trouvé ses fichiers dans `/boot/grub`. C'est tout ce qu'il y a à faire. Quitter le shell **grub** :

**quit**

Créez un fichier « liste de menus » définissant le menu de démarrage de Grub :

```
cat > /boot/grub/menu.lst << "EOF"
# Début de /boot/grub/menu.lst

# Par défaut, lance la première entrée du menu.
default 0

# Attend 30 secondes avant de lancer le noyau par défaut.
timeout 30

# Utilise de jolies couleurs.
color green/black light-green/black

# La première entrée concerne LFS.
title LFS 6.1.1
root (hd0,3)
kernel /boot/lfskernel-2.6.11.12 root=/dev/hda4
EOF
```

Ajoutez une entrée pour votre distribution hôte si vous le souhaitez. Cela pourrait ressembler à ceci :

```
cat >> /boot/grub/menu.lst << "EOF"
title Red Hat
root (hd0,2)
kernel /boot/kernel-2.6.5 root=/dev/hda3
initrd /boot/initrd-2.6.5
EOF
```

Dans le cas d'une machine avec plusieurs systèmes d'exploitation, l'entrée suivante devrait le permettre :

```
cat >> /boot/grub/menu.lst << "EOF"
title Windows
rootnoverify (hd0,0)
chainloader +1
EOF
```

Si **info grub** ne fournit pas toutes les données nécessaires, plus d'informations concernant Grub sont disponibles sur le site web, situé sur <http://www.gnu.org/software/grub/>.

Le FHS stipule que le fichier menu.lst de GRUB doit être un lien symbolique vers /etc/grub/menu.lst. Pour satisfaire ce pré-requis, lancez la commande suivante :

```
mkdir -v /etc/grub &&
ln -sv /boot/grub/menu.lst /etc/grub
```

# Chapitre 9. Fin

## 9.1. La fin

Bien joué ! Le nouveau système LFS est installé. Nous vous souhaitons de bien vous amuser avec votre nouveau système Linux rutilant.

Une bonne idée serait de créer un fichier `/etc/lfs-release`. Avec ce fichier, il vous est très facile (ainsi que pour nous si vous avez besoin de demander de l'aide) de savoir quelle version de LFS vous avez installée sur votre système. Créez ce fichier en lançant :

```
echo 6.1.1 > /etc/lfs-release
```

## 9.2. Enregistrez-vous

Maintenant que vous avez terminé le livre, voulez-vous être enregistré comme utilisateur de LFS ? Allez directement sur <http://www.linuxfromscratch.org/cgi-bin/lfscounter.cgi> et enregistrez-vous comme utilisateur LFS en entrant votre nom et la première version de LFS que vous avez utilisée.

Redémarrons et replongeons maintenant dans LFS.

## 9.3. Redémarrer le système

Maintenant que tous les logiciels ont été installés, il est temps de redémarrer votre ordinateur. Néanmoins, vous devez savoir certaines choses. Le système que vous avez créé dans ce livre est vraiment minimal et a toutes les chances de ne pas avoir les fonctionnalités dont vous aurez besoin pour continuer. En installant quelques autres paquetages à partir du livre BLFS en restant dans l'environnement chroot actuel, vous serez dans une bien meilleure position pour continuer une fois que vous aurez redémarré votre nouvelle installation LFS. Installer un navigateur web en mode texte, comme Lynx, vous permettra de lire facilement le livre BLFS dans un terminal virtuel tout en construisant des paquetages dans un autre. Le paquetage 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 paquetages comme Dhcpcd ou PPP pourrait aussi être utile.

Maintenant que nous vous avons prévenu, démarrons notre toute nouvelle installation LFS pour la première fois ! Tout d'abord, quittez l'environnement chroot :

```
logout
```

Puis, démontez les systèmes de fichiers virtuels :

```
umount -v $LFS/dev/pts  
umount -v $LFS/dev/shm  
umount -v $LFS/dev  
umount -v $LFS/proc  
umount -v $LFS/sys
```

Démontez le système de fichiers LFS :

```
umount -v $LFS
```

Si plusieurs partitions ont été créées, démontez les autres partitions avant de démonter la principale, comme ceci :

```
umount -v $LFS/usr  
umount -v $LFS/home  
umount -v $LFS
```

Maintenant, redémarrez le système avec :

```
shutdown -r now
```

En supposant que le chargeur de démarrage Grub a été initialisé comme indiqué plus tôt, le menu est préparé pour démarrer *LFS 6.1.1* automatiquement.

À la fin du redémarrage, le système LFS est prêt à être utilisé et des logiciels peuvent enfin être installés pour satisfaire vos besoins.

## 9.4. Et maintenant?

Merci d'avoir lu le livre LFS. Nous espérons que vous avez trouvé ce livre utile et que vous avez appris des choses sur la création d'un système.

Maintenant que le système LFS est installé, vous êtes peut-être en train de vous demander « Et maintenant ? » 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 :

- Freshmeat.net (<http://freshmeat.net/>)

Freshmeat peut vous prévenir (par email) des nouvelles versions des 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>.

- Beyond Linux From Scratch

Le livre Beyond Linux From Scratch couvre les procédures d'installation d'un grand nombre de logiciels en dehors du livre LFS. Le projet BLFS est disponible sur <http://www.linuxfromscratch.org/blfs/>.

- Astuces LFS (LFS Hints)

Les astuces LFS sont une collection de documents éducatifs soumis par des volontaires à la communauté LFS. Ces astuces sont disponibles sur <http://www.linuxfromscratch.org/hints/list.html>.

- Listes de diffusion

Il existe plusieurs listes de diffusion 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/>.

# Annexes



# Annexe A. Acronymes et termes

<b>ABI</b>	Application Binary Interface
<b>ALFS</b>	Automated Linux From Scratch
<b>ALSA</b>	Advanced Linux Sound Architecture
<b>API</b>	Application Programming Interface
<b>ASCII</b>	American Standard Code for Information Interchange
<b>BIOS</b>	Basic Input/Output System
<b>BLFS</b>	Beyond Linux From Scratch
<b>BSD</b>	Berkeley Software Distribution
<b>chroot</b>	change root
<b>CMOS</b>	Complementary Metal Oxide Semiconductor
<b>COS</b>	Class Of Service
<b>CPU</b>	Central Processing Unit
<b>CRC</b>	Cyclic Redundancy Check
<b>CVS</b>	Concurrent Versions System
<b>DHCP</b>	Dynamic Host Configuration Protocol
<b>DNS</b>	Domain Name Service
<b>EGA</b>	Enhanced Graphics Adapter
<b>ELF</b>	Executable and Linkable Format
<b>EOF</b>	End of File
<b>EQN</b>	equation
<b>EVMS</b>	Enterprise Volume Management System
<b>ext2</b>	second extended file system
<b>FAQ</b>	Frequently Asked Questions
<b>FHS</b>	Filesystem Hierarchy Standard
<b>FIFO</b>	First-In, First Out
<b>FQDN</b>	Fully Qualified Domain Name
<b>FTP</b>	File Transfer Protocol
<b>GB</b>	Gibabytes
<b>GCC</b>	GNU Compiler Collection
<b>GID</b>	Group Identifier
<b>GMT</b>	Greenwich Mean Time

<b>GPG</b>	GNU Privacy Guard
<b>HTML</b>	Hypertext Markup Language
<b>IDE</b>	Integrated Drive Electronics
<b>IEEE</b>	Institute of Electrical and Electronic Engineers
<b>IO</b>	Input/Output
<b>IP</b>	Internet Protocol
<b>IPC</b>	Inter-Process Communication
<b>IRC</b>	Internet Relay Chat
<b>ISO</b>	International Organization for Standardization
<b>ISP</b>	Internet Service Provider
<b>KB</b>	Kilobytes
<b>LED</b>	Light Emitting Diode
<b>LFS</b>	Linux From Scratch
<b>LSB</b>	Linux Standards Base
<b>MB</b>	Megabytes
<b>MBR</b>	Master Boot Record
<b>MD5</b>	Message Digest 5
<b>NIC</b>	Network Interface Card
<b>NLS</b>	Native Language Support
<b>NNTP</b>	Network News Transport Protocol
<b>NPTL</b>	Native POSIX Threading Library
<b>OSS</b>	Open Sound System
<b>PCH</b>	Pre-Compiled Headers
<b>PCRE</b>	Perl Compatible Regular Expression
<b>PID</b>	Process Identifier
<b>PLFS</b>	Pure Linux From Scratch
<b>PTY</b>	pseudo terminal
<b>QA</b>	Quality Assurance
<b>QOS</b>	Quality Of Service
<b>RAM</b>	Random Access Memory
<b>RPC</b>	Remote Procedure Call
<b>RTC</b>	Real Time Clock
<b>SBU</b>	Standard Build Unit
<b>SCO</b>	The Santa Cruz Operation

<b>SGR</b>	Select Graphic Rendition
<b>SHA1</b>	Secure-Hash Algorithm 1
<b>SMP</b>	Symmetric Multi-Processor
<b>TLDP</b>	The Linux Documentation Project
<b>TFTP</b>	Trivial File Transfer Protocol
<b>TLS</b>	Thread-Local Storage
<b>UID</b>	User Identifier
<b>umask</b>	user file-creation mask
<b>USB</b>	Universal Serial Bus
<b>UTC</b>	Coordinated Universal Time
<b>UUID</b>	Universally Unique Identifier
<b>VC</b>	Virtual Console
<b>VGA</b>	Video Graphics Array
<b>VT</b>	Virtual Terminal

# Annexe B. Remerciements

Nous aimerions remercier les personnes et organisations suivantes pour leur contributions au projet Linux From Scratch.

- *Gerard Beekmans* <gerard@linuxfromscratch.org> – créateur de LFS, leader du projet
- *Matthew Burgess* <matthew@linuxfromscratch.org> – leader du projet LFS, rédacteur technique LFS, gestionnaire des versions de LFS
- *Archaic* <archaic@linuxfromscratch.org> – rédacteur technique LFS, leader du projet HLFS, rédacteur BLFS, mainteneur des projets Astuces et Correctifs
- *Nathan Coulson* <nathan@linuxfromscratch.org> – Mainteneur de LFS-Bootscripts
- *Bruce Dubbs* <bdubbs@linuxfromscratch.org> – leader du projet BLFS
- *Manuel Canales Esparcia* <manuel@linuxfromscratch.org> – mainteneur des fichiers XML et XSL pour LFS/BLFS/HLFS
- *Jim Gifford* <jim@linuxfromscratch.org> – rédacteur technique LFS, leader du projet Correctifs
- *Jeremy Huntwork* <jhuntwork@linuxfromscratch.org> – rédacteur technique LFS, mainteneur du LiveCD LFS, leader du projet ALFS
- *Anderson Lizardo* <lizardo@linuxfromscratch.org> – mainteneur des scripts du site web
- *Ryan Oliver* <ryan@linuxfromscratch.org> – mainteneur des outils de production LFS
- *James Robertson* <jwrober@linuxfromscratch.org> – mainteneur de Bugzilla
- *Tushar Teredesai* <tushar@linuxfromscratch.org> – rédacteur en chef de BLFS, leader des projets Astuces et Correctifs

## Traducteurs

- *Manuel Canales Esparcia* <macana@lfs-es.org> – projet de traduction espagnol
- *Johan Lenglet* <johan@linuxfromscratch.org> – projet de traduction français
- *Anderson Lizardo* <lizardo@linuxfromscratch.org> – projet de traduction portugais
- *Thomas Reitelbach* <tr@erdfunkstelle.de> – projet de traduction allemand

## Mainteneurs des miroirs

### Miroirs d'Amérique du Nord

- *Scott Kveton* <scott@osuosl.org> – miroir lfs.oregonstate.edu
- *Mikhail Pastukhov* <miha@xuy.biz> – miroir lfs.130th.net
- *William Astle* <lost@l-w.net> – miroir ca.linuxfromscratch.org
- *Jeremy Polen* <jpolen@rackspace.com> – miroir us2.linuxfromscratch.org
- *Tim Jackson* <tim@idge.net> – miroir linuxfromscratch.idge.net

- *Jeremy Utley* <jeremy@linux-phreak.net> – miroir lfs.linux-phreak.net

## Miroirs d'Amérique du Sud

- *Manuel Canales Esparcia* <manuel@linuxfromscratch.org> – miroir lfsmirror.lfs-es.org
- *Andres Meggiotto* <sysop@mesi.com.ar> – miroir lfs.mesi.com.ar
- *Eduardo B. Fonseca* <ebf@aedsolucoes.com.br> – miroir br.linuxfromscratch.org

## Miroirs européens

- *Barna Koczka* <barna@siker.hu> – miroir hu.linuxfromscratch.org
- *UK Mirror Service* – miroir linuxfromscratch.mirror.ac.uk
- *Martin Voss* <Martin.Voss@ada.de> – miroir lfs.linux-matrix.net
- *Guido Passet* <guido@primerelay.net> – miroir nl.linuxfromscratch.org
- *Bastiaan Jacques* <baafie@planet.nl> – miroir lfs.pagefault.net
- *Roel Neefs* <lfs-mirror@linuxfromscratch.rave.org> – miroir linuxfromscratch.rave.org
- *Justin Knierim* <justin@jrknierim.de> – miroir www.lfs-matrix.de
- *Stephan Brendel* <stevie@stevie20.de> – miroir lfs.netservice-neuss.de
- *Antonin Sprinzl* <Antonin.Sprinzl@tuwien.ac.at> – miroir at.linuxfromscratch.org
- *Fredrik Danerklint* <fredan-lfs@fredan.org> – miroir se.linuxfromscratch.org
- *Parisian sysadmins* <archive@doc.cs.univ-paris8.fr> – miroir www2.fr.linuxfromscratch.org
- *Alexander Velin* <velin@zadnik.org> – miroir bg.linuxfromscratch.org
- *Dirk Webster* <dirk@securewebservices.co.uk> – miroir lfs.securewebservices.co.uk
- *Thomas Skyt* <thomas@sofagang.dk> – miroir dk.linuxfromscratch.org
- *Simon Nicoll* <sime@dot-sime.com> – miroir uk.linuxfromscratch.org

## Miroirs Asiatiques

- *Pui Yong* <pyng@spam.averse.net> – miroir sg.linuxfromscratch.org
- *Stuart Harris* <stuart@althalus.me.uk> – miroir lfs.mirror.intermedia.com.sg

## Miroirs australiens

- *Jason Andrade* <jason@dstc.edu.au> – miroir au.linuxfromscratch.org

## Anciens membres de l'équipe du projet

- *Christine Barczak* <theladyskye@linuxfromscratch.org> – rédactrice LFS
- Timothy Bauscher
- Robert Briggs
- Ian Chilton
- *Jeroen Coumans* <jeroen@linuxfromscratch.org> – développeur du site web, mainteneur de la FAQ
- Alex Groenewoud – rédacteur technique LFS
- Marc Heerdink
- Mark Hymers
- Seth W. Klein – mainteneur de la FAQ
- *Nicholas Leippe* <nicholas@linuxfromscratch.org> – mainteneur du Wiki
- Simon Perreault
- *Scot Mc Pherson* <scot@linuxfromscratch.org> – mainteneur de la passerelle NNTP LFS
- *Alexander Patrakov* <semzx@newmail.ru> – rédacteur technique LFS
- *Greg Schafer* <gschafer@zip.com.au> – rédacteur technique LFS
- Jesse Tie-Ten-Quee – rédacteur technique LFS
- *Jeremy Utley* <jeremy@linuxfromscratch.org> – rédacteur technique LFS, mainteneur de Bugzilla, mainteneur des LFS-Bootscripts
- *Zack Winkles* <zwinkles@gmail.com> – rédacteur technique LFS

## Un remerciement tout particulier à nos donateurs

- *Dean Benson* <dean@vipersoft.co.uk> pour plusieurs contributions financières
- *Hagen Herrschaft* <hrx@hrxnet.de> pour le don d'un système P4 2.2 GHz, fonctionnant actuellement sous le nom de Lorien
- *VA Software* qui, sous le titre de *Linux.com*, a donné une station de travail VA Linux 420 (ancien StartX SP2)

- Mark Stone pour le don de Belgarath, le serveur [linuxfromscratch.org](http://linuxfromscratch.org)

# Index

## Paquets

Autoconf: 156  
 Automake: 158  
 Bash: 160  
   outils: 83  
 Binutils: 112  
   outils, passe 1: 49  
   outils, passe 2: 67  
 Bison: 138  
   outils: 85  
 Scripts de démarrage: 208  
   usage: 210  
 Bzip2: 164  
   outils: 71  
 Coreutils: 118  
   outils: 70  
 DejaGNU: 63  
 Diffutils: 166  
   outils: 73  
 E2fsprogs: 169  
 Expect: 61  
 File: 162  
 Findutils: 127  
   outils: 74  
 Flex: 144  
   outils: 86  
 Gawk: 129  
   outils: 69  
 GCC: 115  
   outils, passe 1: 51  
   outils, passe 2: 64  
 Gettext: 146  
   outils: 78  
 Glibc: 103  
   outils: 54  
 Grep: 172  
   outils: 76  
 Groff: 140  
 Grub: 173  
   configuration: 234  
 Gzip: 175  
   tools: 72  
 Hotplug: 177  
 Iana-Etc: 126  
 Inetutils: 148  
 IPRoute2: 150  
 Kbd: 167  
 Less: 139  
 Libtool: 163

Linux: 231  
 Linux-Libc-Headers: 101  
   outils, en-têtes: 53  
 M4: 137  
   outils: 84  
 Make: 181  
   outils: 75  
 Man: 179  
 Man-pages: 102  
 Mktemp: 125  
 Module-Init-Tools: 182  
 Ncurses: 130  
   outils: 79  
 Patch: 184  
   outils: 80  
 Perl: 152  
   outils: 88  
 Procps: 185  
 Psmisc: 187  
 Readline: 132  
 Sed: 143  
   outils: 77  
 Shadow: 188  
   configuration: 190  
 Sysklogd: 192  
   configuration: 193  
 Sysvinit: 194  
   configuration: 194  
 Tar: 197  
   outils: 81  
 Tcl: 59  
 Texinfo: 154  
   outils: 82  
 Udev: 198  
   usage: 212  
 Util-linux: 200  
   outils: 87  
 Vim: 134  
 Zlib: 123

## Programmes

a2p: 152 , 152  
 acinstall: 158 , 158  
 aclocal: 158 , 158  
 aclocal-1.9.5: 158 , 158  
 addftinfo: 140 , 140  
 addr2line: 112 , 113  
 afmtodit: 140 , 140  
 agetty: 200 , 201  
 apropos: 179 , 180  
 ar: 112 , 113  
 arch: 200 , 201



as: 112 , 113  
 autoconf: 156 , 156  
 autoheader: 156 , 156  
 autom4te: 156 , 156  
 automake: 158 , 158  
 automake-1.9.5: 158 , 158  
 autopoint: 146 , 146  
 autoreconf: 156 , 156  
 autoscan: 156 , 156  
 autoupdate: 156 , 156  
 awk: 129 , 129  
 badblocks: 169 , 170  
 basename: 118 , 119  
 bash: 160 , 161  
 bashbug: 160 , 161  
 bigram: 127 , 127  
 bison: 138 , 138  
 blkid: 169 , 170  
 blockdev: 200 , 201  
 bunzip2: 164 , 165  
 bzip2: 164 , 165  
 bzcat: 164 , 165  
 bzcmp: 164 , 165  
 bzdiff: 164 , 165  
 bzegrep: 164 , 165  
 bzfgrep: 164 , 165  
 bzgrep: 164 , 165  
 bzip2: 164 , 165  
 bzip2recover: 164 , 165  
 bzless: 164 , 165  
 bzip2more: 164 , 165  
 c++: 115 , 116  
 c++filt: 112 , 113  
 c2ph: 152 , 152  
 cal: 200 , 201  
 captinfo: 130 , 131  
 cat: 118 , 119  
 catchsegv: 103 , 107  
 cc: 115 , 116  
 cfdisk: 200 , 201  
 chage: 188 , 190  
 chattr: 169 , 170  
 chfn: 188 , 190  
 chgrp: 118 , 119  
 chkdupexe: 200 , 201  
 chmod: 118 , 119  
 chown: 118 , 119  
 chpasswd: 188 , 190  
 chroot: 118 , 119  
 chsh: 188 , 190  
 chvt: 167 , 167  
 cksum: 118 , 119  
 clear: 130 , 131  
 cmp: 166 , 166  
 code: 127 , 127  
 col: 200 , 201  
 colcrt: 200 , 201  
 colrm: 200 , 201  
 column: 200 , 201  
 comm: 118 , 119  
 compile: 158 , 158  
 compile\_et: 169 , 170  
 compress: 175 , 175  
 config.charset: 146 , 146  
 config.guess: 158 , 158  
 config.rpath: 146 , 146  
 config.sub: 158 , 158  
 cp: 118 , 119  
 cpp: 115 , 116  
 csplit: 118 , 119  
 ctrlaltdel: 200 , 201  
 ctstat: 150 , 150  
 cut: 118 , 119  
 cytune: 200 , 201  
 date: 118 , 119  
 dd: 118 , 119  
 ddate: 200 , 201  
 dealloct: 167 , 167  
 debugfs: 169 , 170  
 depcomp: 158 , 159  
 depmod: 182 , 182  
 df: 118 , 120  
 diff: 166 , 166  
 diff3: 166 , 166  
 dir: 118 , 120  
 dircolors: 118 , 120  
 dirname: 118 , 120  
 dmesg: 200 , 201  
 dprofpp: 152 , 152  
 du: 118 , 120  
 dumpe2fs: 169 , 170  
 dumpkeys: 167 , 167  
 e2fsck: 169 , 170  
 e2image: 169 , 170  
 e2label: 169 , 170  
 echo: 118 , 120  
 efm\_filter.pl: 134 , 135  
 efm\_perl.pl: 134 , 135  
 egrep: 172 , 172  
 elisp-comp: 158 , 159  
 elvtune: 200 , 201  
 en2exs: 152 , 153  
 env: 118 , 120  
 envsubst: 146 , 146  
 eqn: 140 , 140

eqn2graph: 140 , 140  
 ex: 134 , 136  
 expand: 118 , 120  
 expect: 61 , 62  
 expiry: 188 , 190  
 expr: 118 , 120  
 factor: 118 , 120  
 faillog: 188 , 190  
 false: 118 , 120  
 fdformat: 200 , 201  
 fdisk: 200 , 201  
 fgconsole: 167 , 167  
 fgrep: 172 , 172  
 file: 162 , 162  
 find: 127 , 127  
 find2perl: 152 , 153  
 findfs: 169 , 170  
 flex: 144 , 145  
 fmt: 118 , 120  
 fold: 118 , 120  
 frcode: 127 , 127  
 free: 185 , 185  
 fsck: 169 , 170  
 fsck.cramfs: 200 , 201  
 fsck.ext2: 169 , 170  
 fsck.ext3: 169 , 170  
 fsck.minix: 200 , 201  
 ftp: 148 , 149  
 fuser: 187 , 187  
 g++: 115 , 116  
 gawk: 129 , 129  
 gawk-3.1.4: 129 , 129  
 gcc: 115 , 116  
 gccbug: 115 , 116  
 gcov: 115 , 117  
 gencat: 103 , 107  
 geqn: 140 , 140  
 getconf: 103 , 107  
 getent: 103 , 107  
 getkeycodes: 167 , 167  
 getopt: 200 , 201  
 gettext: 146 , 146  
 gettextize: 146 , 146  
 getunimap: 167 , 167  
 gpasswd: 188 , 191  
 gprof: 112 , 113  
 grcat: 129 , 129  
 grep: 172 , 172  
 grn: 140 , 140  
 grodvi: 140 , 141  
 groff: 140 , 141  
 groffer: 140 , 141  
 grog: 140 , 141  
 grolbp: 140 , 141  
 grolj4: 140 , 141  
 grops: 140 , 141  
 grotty: 140 , 141  
 groupadd: 188 , 191  
 groupdel: 188 , 191  
 groupmod: 188 , 191  
 groups: 118 , 120  
 grpck: 188 , 191  
 grpconv: 188 , 191  
 grpunconv: 188 , 191  
 grub: 173 , 173  
 grub-install: 173 , 173  
 grub-md5-crypt: 173 , 173  
 grub-terminfo: 173 , 173  
 gtbl: 140 , 141  
 gunzip: 175 , 175  
 gzexe: 175 , 175  
 gzip: 175 , 175  
 h2ph: 152 , 153  
 h2xs: 152 , 153  
 halt: 194 , 196  
 head: 118 , 120  
 hexdump: 200 , 201  
 hostid: 118 , 120  
 hostname: 118 , 120  
 hostname: 146 , 146  
 hotplug: 177 , 178  
 hpftodit: 140 , 141  
 hwclock: 200 , 201  
 iconv: 103 , 107  
 iconvconfig: 103 , 107  
 id: 118 , 120  
 ifcfg: 150 , 150  
 ifnames: 156 , 157  
 ifstat: 150 , 150  
 igawk: 129 , 129  
 indxbib: 140 , 141  
 info: 154 , 155  
 infocmp: 130 , 131  
 infokey: 154 , 155  
 infotocap: 130 , 131  
 init: 194 , 196  
 insmod: 182 , 183  
 insmod.static: 182 , 183  
 install: 118 , 120  
 install-info: 154 , 155  
 install-sh: 158 , 159  
 ip: 150 , 150  
 ipcrm: 200 , 201  
 ipcs: 200 , 202

isosize: 200 , 202  
 join: 118 , 120  
 kbdrate: 167 , 167  
 kbd\_mode: 167 , 167  
 noyau: 231 , 233  
 kill: 185 , 185  
 killall: 187 , 187  
 killall5: 194 , 196  
 klogd: 192 , 193  
 last: 194 , 196  
 lastb: 194 , 196  
 lastlog: 188 , 191  
 ld: 112 , 113  
 ldconfig: 103 , 107  
 ldd: 103 , 107  
 lddlibc4: 103 , 107  
 less: 139 , 139  
 less.sh: 134 , 136  
 lessecho: 139 , 139  
 lesskey: 139 , 139  
 lex: 144 , 145  
 libnetcfg: 152 , 153  
 libtool: 163 , 163  
 libtoolize: 163 , 163  
 line: 200 , 202  
 link: 118 , 120  
 lkbib: 140 , 141  
 ln: 118 , 120  
 lnstat: 150 , 151  
 loadkeys: 167 , 167  
 loadunimap: 167 , 167  
 locale: 103 , 107  
 localedef: 103 , 107  
 locate: 127 , 127  
 logger: 200 , 202  
 login: 188 , 191  
 logname: 118 , 120  
 logoutd: 188 , 191  
 logsave: 169 , 170  
 look: 200 , 202  
 lookbib: 140 , 141  
 losetup: 200 , 202  
 ls: 118 , 120  
 lsattr: 169 , 170  
 lsmod: 182 , 183  
 m4: 137 , 137  
 make: 181 , 181  
 makeinfo: 154 , 155  
 makewhatis: 179 , 180  
 man: 179 , 180  
 man2dvi: 179 , 180  
 man2html: 179 , 180  
 mapscrn: 167 , 168  
 mbchk: 173 , 174  
 mcookie: 200 , 202  
 md5sum: 118 , 120  
 mdate-sh: 158 , 159  
 mesg: 194 , 196  
 missing: 158 , 159  
 mkdir: 118 , 120  
 mke2fs: 169 , 171  
 mkfifo: 118 , 120  
 mkfs: 200 , 202  
 mkfs.bfs: 200 , 202  
 mkfs.cramfs: 200 , 202  
 mkfs.ext2: 169 , 171  
 mkfs.ext3: 169 , 171  
 mkfs.minix: 200 , 202  
 mkinstalldirs: 158 , 159  
 mklost+found: 169 , 171  
 mknod: 118 , 120  
 mkpasswd: 188 , 191  
 mkswap: 200 , 202  
 mktemp: 125 , 125  
 mk\_cmds: 169 , 170  
 mmroff: 140 , 141  
 modinfo: 182 , 183  
 modprobe: 182 , 183  
 more: 200 , 202  
 mount: 200 , 202  
 mountpoint: 194 , 196  
 msgattrib: 146 , 147  
 msgcat: 146 , 147  
 msgcmp: 146 , 147  
 msgcomm: 146 , 147  
 msgconv: 146 , 147  
 msgen: 146 , 147  
 msgexec: 146 , 147  
 msgfilter: 146 , 147  
 msgfmt: 146 , 147  
 msggrep: 146 , 147  
 msginit: 146 , 147  
 msgmerge: 146 , 147  
 msgunfmt: 146 , 147  
 msguniq: 146 , 147  
 mtrace: 103 , 108  
 mv: 118 , 120  
 mve.awk: 134 , 136  
 namei: 200 , 202  
 neqn: 140 , 141  
 newgrp: 188 , 191  
 newusers: 188 , 191  
 ngettext: 146 , 147  
 nice: 118 , 121

nl: 118 , 121  
 nm: 112 , 113  
 nohup: 118 , 121  
 nroff: 140 , 141  
 nscd: 103 , 108  
 nscd\_nischeck: 103 , 108  
 nstat: 150 , 151  
 objcopy: 112 , 113  
 objdump: 112 , 113  
 od: 118 , 121  
 openvt: 167 , 168  
 passwd: 188 , 191  
 paste: 118 , 121  
 patch: 184 , 184  
 pathchk: 118 , 121  
 pcprofiledump: 103 , 108  
 perl: 152 , 153  
 perl5.8.7: 152 , 153  
 perlbug: 152 , 153  
 perlcc: 152 , 153  
 perldoc: 152 , 153  
 perlivp: 152 , 153  
 pfbtops: 140 , 141  
 pg: 200 , 202  
 pgawk: 129 , 129  
 pgawk-3.1.4: 129 , 129  
 pgrep: 185 , 185  
 pic: 140 , 141  
 pic2graph: 140 , 141  
 piconv: 152 , 153  
 pidof: 194 , 196  
 ping: 148 , 149  
 pinky: 118 , 121  
 pivot\_root: 200 , 202  
 pkill: 185 , 185  
 pl2pm: 152 , 153  
 pltags.pl: 134 , 136  
 pmap: 185 , 185  
 pod2html: 152 , 153  
 pod2latex: 152 , 153  
 pod2man: 152 , 153  
 pod2text: 152 , 153  
 pod2usage: 152 , 153  
 podchecker: 152 , 153  
 podselect: 152 , 153  
 post-grohtml: 140 , 141  
 poweroff: 194 , 196  
 pr: 118 , 121  
 pre-grohtml: 140 , 141  
 printenv: 118 , 121  
 printf: 118 , 121  
 ps: 185 , 185  
 psed: 152 , 153  
 psfaddtable: 167 , 168  
 psfgettable: 167 , 168  
 psfstriptime: 167 , 168  
 psfxtable: 167 , 168  
 pstree: 187 , 187  
 pstree.x11: 187 , 187  
 pstruct: 152 , 153  
 ptx: 118 , 121  
 pt\_chown: 103 , 108  
 pwcat: 129 , 129  
 pwck: 188 , 191  
 pwconv: 188 , 191  
 pwd: 118 , 121  
 pwunconv: 188 , 191  
 py-compile: 158 , 159  
 ramsize: 200 , 202  
 ranlib: 112 , 113  
 raw: 200 , 202  
 rcp: 148 , 149  
 rdev: 200 , 202  
 readelf: 112 , 113  
 readlink: 118 , 121  
 readprofile: 200 , 202  
 reboot: 194 , 196  
 ref: 134 , 136  
 refer: 140 , 141  
 rename: 200 , 202  
 renice: 200 , 202  
 reset: 130 , 131  
 resize2fs: 169 , 171  
 resizecons: 167 , 168  
 rev: 200 , 202  
 rlogin: 148 , 149  
 rm: 118 , 121  
 rmdir: 118 , 121  
 rmmod: 182 , 183  
 rmt: 197 , 197  
 rootflags: 200 , 202  
 routef: 150 , 151  
 routel: 150 , 151  
 rpcgen: 103 , 108  
 rpcinfo: 103 , 108  
 rsh: 148 , 149  
 rtacct: 150 , 151  
 rtmon: 150 , 151  
 rtpr: 150 , 151  
 rtstat: 150 , 151  
 runlevel: 194 , 196  
 runtest: 63 , 63  
 rview: 134 , 136  
 rvim: 134 , 136

s2p: 152 , 153  
script: 200 , 202  
sdiff: 166 , 166  
sed: 143 , 143  
seq: 118 , 121  
setfdprm: 200 , 202  
setfont: 167 , 168  
setkeycodes: 167 , 168  
setleds: 167 , 168  
setlogcons: 167 , 168  
setmetamode: 167 , 168  
setsid: 200 , 202  
setterm: 200 , 202  
setvesablank: 167 , 168  
sfdisk: 200 , 203  
sg: 188 , 191  
sh: 160 , 161  
sha1sum: 118 , 121  
showconsolefont: 167 , 168  
showkey: 167 , 168  
shred: 118 , 121  
shtags.pl: 134 , 136  
shutdown: 194 , 196  
size: 112 , 113  
skill: 185 , 185  
sleep: 118 , 121  
sln: 103 , 108  
snice: 185 , 185  
soelim: 140 , 141  
sort: 118 , 121  
splain: 152 , 153  
split: 118 , 121  
sprof: 103 , 108  
ss: 150 , 151  
stat: 118 , 121  
strings: 112 , 113  
strip: 112 , 114  
stty: 118 , 121  
su: 188 , 191  
sulogin: 194 , 196  
sum: 118 , 121  
swapdev: 200 , 203  
swapoff: 200 , 203  
swapon: 200 , 203  
symlink-tree: 158 , 159  
sync: 118 , 121  
sysctl: 185 , 185  
syslogd: 192 , 193  
tac: 118 , 121  
tack: 130 , 131  
tail: 118 , 121  
talk: 148 , 149  
tar: 197 , 197  
tbl: 140 , 142  
tc: 150 , 151  
tclsh: 59 , 60  
tclsh8.4: 59 , 60  
tcltags: 134 , 136  
tee: 118 , 121  
telinit: 194 , 196  
telnet: 148 , 149  
tempfile: 125 , 125  
test: 118 , 122  
texi2dvi: 154 , 155  
texi2pdf: 154 , 155  
texindex: 154 , 155  
tfmtodit: 140 , 142  
tftp: 148 , 149  
tic: 130 , 131  
tload: 185 , 185  
toe: 130 , 131  
top: 185 , 185  
touch: 118 , 122  
tput: 130 , 131  
tr: 118 , 122  
troff: 140 , 142  
true: 118 , 122  
tset: 130 , 131  
tsort: 118 , 122  
tty: 118 , 122  
tune2fs: 169 , 171  
tunelp: 200 , 203  
tzselect: 103 , 108  
udev: 198 , 198  
udev: 198 , 199  
udevinfo: 198 , 199  
udevsend: 198 , 199  
udevstart: 198 , 199  
udevtest: 198 , 199  
ul: 200 , 203  
umount: 200 , 203  
uname: 118 , 122  
uncompress: 175 , 176  
unexpand: 118 , 122  
unicode\_start: 167 , 168  
unicode\_stop: 167 , 168  
uniq: 118 , 122  
unlink: 118 , 122  
updatedb: 127 , 127  
uptime: 185 , 185  
useradd: 188 , 191  
userdel: 188 , 191  
usermod: 188 , 191  
users: 118 , 122

utmpdump: 194 , 196  
 uuidgen: 169 , 171  
 vdir: 118 , 122  
 vi: 134 , 136  
 vidmode: 200 , 203  
 view: 134 , 136  
 vigr: 188 , 191  
 vim: 134 , 136  
 vim132: 134 , 136  
 vim2html.pl: 134 , 136  
 vimdiff: 134 , 136  
 vimmm: 134 , 136  
 vimspell.sh: 134 , 136  
 vimtutor: 134 , 136  
 vipw: 188 , 191  
 vmstat: 185 , 186  
 w: 185 , 186  
 wall: 194 , 196  
 watch: 185 , 186  
 wc: 118 , 122  
 whatis: 179 , 180  
 whereis: 200 , 203  
 who: 118 , 122  
 whoami: 118 , 122  
 write: 200 , 203  
 xargs: 127 , 128  
 xgettext: 146 , 147  
 xsubpp: 152 , 153  
 xtrace: 103 , 108  
 xxd: 134 , 136  
 yacc: 138 , 138  
 yes: 118 , 122  
 ylwrap: 158 , 159  
 zcat: 175 , 176  
 zcmp: 175 , 176  
 zdiff: 175 , 176  
 zdump: 103 , 108  
 zegrep: 175 , 176  
 zfgrep: 175 , 176  
 zforce: 175 , 176  
 zgrep: 175 , 176  
 zic: 103 , 108  
 zless: 175 , 176  
 zmore: 175 , 176  
 znew: 175 , 176  
 zsoelim: 140 , 142

## Bibliothèques

ld.so: 103 , 108  
 libanl: 103 , 108  
 libasprintf: 146 , 147  
 libbfd: 112 , 114

libblkid: 169 , 171  
 libBrokenLocale: 103 , 108  
 libbsd-compat: 103 , 108  
 libbz2\*: 164 , 165  
 libc: 103 , 108  
 libcom\_err: 169 , 171  
 libcrypt: 103 , 108  
 libcurses: 130 , 131  
 libdl: 103 , 108  
 libe2p: 169 , 171  
 libexpect-5.43: 61 , 62  
 libext2fs: 169 , 171  
 libfl.a: 144 , 145  
 libform: 130 , 131  
 libg: 103 , 108  
 libgcc\*: 115 , 117  
 libgettextlib: 146 , 147  
 libgettextpo: 146 , 147  
 libgettextsrc: 146 , 147  
 libhistory: 132 , 133  
 libiberty: 112 , 114  
 libieee: 103 , 108  
 libltdl: 163 , 163  
 libm: 103 , 108  
 libmagic: 162 , 162  
 libmcheck: 103 , 108  
 libmemusage: 103 , 108  
 libmenu: 130 , 131  
 libncurses: 130 , 131  
 libnsl: 103 , 108  
 libnss: 103 , 108  
 libopcodes: 112 , 114  
 libpanel: 130 , 131  
 libpcprofile: 103 , 109  
 libproc: 185 , 186  
 libpthread: 103 , 109  
 libreadline: 132 , 133  
 libresolv: 103 , 109  
 librpcsvc: 103 , 109  
 librt: 103 , 109  
 libSegFault: 103 , 108  
 libshadow: 188 , 191  
 libss: 169 , 171  
 libstdc++: 115 , 117  
 libsupc++: 115 , 117  
 libtcl8.4.so: 59 , 60  
 libthread\_db: 103 , 109  
 libutil: 103 , 109  
 libuuid: 169 , 171  
 liby.a: 138 , 138  
 libz: 123 , 124

**Scripts**

/etc/hotplug/\*.agent: 177 , 178  
 /etc/hotplug/\*.rc: 177 , 178  
 checkfs: 208 , 208  
 cleanfs: 208 , 208  
 console: 208 , 208  
     configurer: 217  
 functions: 208 , 208  
 halt: 208 , 208  
 hotplug: 208 , 208  
 ifdown: 208 , 208  
 ifup: 208 , 208  
 localnet: 208 , 208  
     /etc/hosts: 226  
     configuration: 225  
 mountfs: 208 , 208  
 mountkernfs: 208 , 208  
 network: 208 , 208  
     /etc/hosts: 226  
     configurer: 227  
 rc: 208 , 208  
 reboot: 208 , 209  
 sendsignals: 208 , 209  
 setclock: 208 , 209  
     configurer: 216  
 static: 208 , 209  
 swap: 208 , 209  
 sysklogd: 208 , 209  
     configuration: 219  
 template: 208 , 209  
 udev: 208 , 209

**Autres**

/boot/config-2.6.11.12: 231 , 233  
 /boot/System.map-2.6.11.12: 231 , 233  
 /dev/\*: 99  
 /etc/group: 97  
 /etc/hosts: 226  
 /etc/hotplug.d: 177 , 178  
 /etc/hotplug/blacklist: 177 , 178  
 /etc/hotplug/hotplug.functions: 177 , 178  
 /etc/hotplug/usb.usermap: 177 , 178  
 /etc/hotplug/{pci,usb}: 177 , 178  
 /etc/inittab: 194  
 /etc/inputrc: 220  
 /etc/ld.so.conf: 107  
 /etc/limits: 188  
 /etc/localtime: 106  
 /etc/login.access: 188  
 /etc/login.defs: 188  
 /etc/nsswitch.conf: 106

/etc/passwd: 97  
 /etc/profile: 222  
 /etc/protocols: 126  
 /etc/resolv.conf: 227  
 /etc/services: 126  
 /etc/syslog.conf: 193  
 /etc/udev: 198 , 199  
 /etc/vim: 135  
 /lib/firmware: 177 , 178  
 /usr/include/{asm,linux}/\*.h: 101 , 101  
 /var/log/btmp: 97  
 /var/log/hotplug/events: 177 , 178  
 /var/log/lastlog: 97  
 /var/log/wtmp: 97  
 /var/run/utmp: 97  
 pages man: 102 , 102