Linux From Scratch

Version 6.1.1

Gerard Beekmans

Linux From Scratch: Version 6.1.1

par Gerard Beekmans

Table des matières

Préface

1. Avant-propos

Mes aventures avec Linux ont commencé en 1998 lorsque j'ai téléchargé et installé ma première distribution. Après avoir travaillé avec 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, mais chacune avait ses avantages et ses inconvénients. Finalement, j'ai réalisé que si je voulais avoir pleine satisfaction de mon système Linux, je devais créer le mien en partant 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 installerait les 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 je suis resté convaincu qu'un système pourrait être construit pour se conformer à mes besoins et désirs plutôt qu'à un standard qui ne correspondrait pas à ce que je cherchais.

Après avoir résolu 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 minimalistes, plus rapides et moins encombrant que les systèmes d'exploitation traditionnels. J'ai appelé ce système « Linux From Scratch » (Linux à partir de rien), ou LFS pour faire court.

En partageant mes objectifs et mes expériences avec d'autres membres de la communauté Linux, il est apparu qu'il y avait un réel intérêt pour les idées que j'avais mises en avant lors de mes aventures Linux. Ces systèmes LFS personnalisés permettent non seulement de répondre aux besoins des utilisateurs, mais sont aussi une excellente opportunité pour les programmeurs et les administrateurs système qui souhaitent améliorer leurs connaissances sous Linux. De cet intérêt est né le projet « Linux From Scratch ».

Le livre « *Linux From Scratch* » fournit aux lecteurs les informations 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 contrôler 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 pourra choisir spécifier où, pourquoi et comment les programmes sont installés. Ce livre permet aux lecteurs de personnaliser complètement les systèmes Linux suivant leurs besoins et d'avoir plus de contrôle 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

Beaucoup de raisons peuvent vous pousser à lire ce livre, la principale étant de vouloir installer un système Linux à partir du code source. La question qui vient alors à l'esprit est « pourquoi se fatiguer à créer à la main un système Linux en partant de rien, alors qu'il suffit de télécharger une distribution existante ? » C'est une bonne question, suffisamment bonne pour justifier cette section du livre, en fait.

Une des raisons d'être de LFS est d'aider à comprendre de l'intérieur comment fonctionne un système Linux. Construire un système LFS vous montre ce qui "fait tourner Linux", et comment les choses interagissent et s'imbriquent les unes dans les autres. Plus important, vous apprenez à le personnaliser afin qu'il soit à votre goût, et réponde à vos besoins.

Un avantage majeur de LFS est de permettre à l'utilisateur plus de contrôle sur son système, sans avoir à dépendre de l'implémentation créée par un autre. Avec LFS, *vous* êtes le pilote et *vous* décidez de chaque aspect de votre système, de l'organisation des répertoires à la configuration des scripts de démarrage. Vous choisissez également où, pourquoi et comment les programmes sont installés.

Un autre intérêt de LFS est la possibilité de créer un système Linux très compact. Avec une distribution classique, vous installerez souvent beaucoup de programmes qui ne seront jamais utilisés. Ces programmes gaspillent de l'espace disque, et pire, parfois même du temps CPU. 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 distributions existantes. Cela vous semble encore trop ? Certains d'entre nous ont travaillé à créer un système LFS minuscule, et nous avons pu installer un système spécialisé pour faire fonctionner le serveur web Apache sur 8 Mo d'espace disque. Avec plus de dépouillement encore, on peut se contenter de 5 Mo, voire moins. Essayez donc d'en faire autant avec une distribution courante ! Et ce n'est que l'un des avantages à concevoir votre propre système Linux.

Une distribution Linux classique, c'est un peu le hamburger que vous achetez au restaurant fast-food : pratique, mais vous n'avez aucune idée de ce qu'il y a dedans. LFS ne vous donne pas de hamburger, mais la recette pour en faire un, vous permettant au passage de l'inspecter, d'enlever les ingrédients non désirés, et de rajouter ce qui, à votre goût, manque. Une fois satisfait par la recette, aiguisez les couteaux et combinez tout ce petit monde. Vous avez même la chance de pouvoir le cuire comme vous le souhaitez : faites-le griller, frire, au four, au barbecue, ou mangez-le cru.

Une seconde analogie possible 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, les personnalisant suivant les besoins et préférences des utilisateurs.

Vous êtes concerné par la sécurité de votre système ? LFS vous apporte un surcroît de sécurité. En compilant le système complet à partir des sources, vous pouvez tout vérifier si vous le voulez, et 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. A moins d'examiner et d'implémenter vous-même le correctif, vous n'avez même pas la garantie que ce nouveau paquet résolve le problème de manière adéquate.

Comme il a déjà été dit, le but de Linux From Scratch est de construire un système-fondation complet et utilisable. Si vous ne souhaitez pas construire un système complet, vous pourriez ne pas bénéficier de toutes les 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 propose la construction d'un système similaire à celui de ce livre, mais se limite à la création d'un système capable de

démarrer jusqu'à l'invite du BASH. Tout dépend de votre objectif. 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 toutes les lister ici. Cette section n'aborde que la partie visible de l'iceberg. En progressant dans votre expérience de LFS, vous verrez 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 indiquées. En particulier, le lecteur devrait au minimum savoir utiliser la ligne de commande (le shell) pour copier et déplacer des fichiers et des répertoires, pour lister le contenu de répertoires et de fichiers, et pour changer de répertoire. Le lecteur est également supposé avoir une connaissance raisonnable de l'utilisation et l'installation de logiciels Linux.

Comme le livre LFS attend *au moins* de vous ce niveau de connaissance, il est peu probable que les différents forums de support LFS vous fournissent une assistance sur ces questions de base ; il est possible que vos questions restent sans réponses, ou que vous soyez renvoyé à la liste des lectures pre-LFS essentielles.

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 de logiciels Unix « générique » 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.* lire avant tout

  C'est une astuce LFS écrite spécifiquement pour les nouveaux utilisateurs Linux. Elle comprend une liste de liens vers d'excellentes sources d'information sur de nombreux thèmes. Toute personne désirant installer LFS devrait connaître au moins basiquement la majorité des sujets qui y sont abordés.

4. Système hôte requis

L'hôte doit utiliser au minimum un noyau 2.6.2 compilé avec GCC-3.0 ou ultérieur. Il y a deux raisons majeures à cela. D'abord, la suite de tests NPTL (*Native POSIX Threading Library*) causera une erreur de segmentation si le noyau de l'hôte n'a pas été compilé avec GCC-3.0 ou ultérieur. 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` qui n'a été intégré que récemment dans la plupart des pilotes du noyau. Nous voulons être sûrs que tous les périphériques système critiques seront correctement créés.

Pour vérifier si le noyau de l'hôte répond bien aux prérequis indiqués ci-dessus, lancez la commande suivante :

```
cat /proc/version
```

Elle devrait produire une sortie semblable à ceci :

```
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 le noyau de l'hôte n'est pas un 2.6.2 (ou ultérieur) ou qu'il n'a pas été compilé avec GCC-3.0 (ou ultérieur), vous devez installer un noyau adéquat. Il existe deux méthodes pour ceci. Tout d'abord, regardez si votre distribution Linux fournit un noyau 2.6.2 (ou ultérieur). Dans ce cas, vous pouvez l'installer. Si ce n'est pas le cas ou si vous préférez ne pas l'installer, 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 situées dans Chapitre 8. La deuxième option peut aussi être vue comme un test de vos connaissances Linux. Si cette étape vous semble trop compliquée, alors le livre LFS ne vous sera probablement pas d'une grande utilité actuellement.

5. Typographie

Pour que les choses soient plus faciles à suivre, ce livre emploie quelques conventions typographiques. Cette section contient des exemples des formats typographiques rencontrés dans ce livre.

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

Le texte dans ce style doit être tapé sur votre ordinateur exactement comme il apparaît dans le livre, sauf -rare- indication contraire. Ce style est également employé dans les sections d'explications pour identifier la commande dont on parle.

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

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

Emphase

Les utilisations de ce style dans le livre sont variées, mais 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 comme ceux vers 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 que contiennent les lignes suivantes, jusqu'à la séquence de fin de fichier (EOF). Cette section est donc généralement saisie exactement comme elle apparaît dans le livre.

`[TEXTE À REMPLACER]`

Ce style est appliqué à une portion de texte qui ne doit pas être saisie telle quelle, ni copiée/collée.

`passwd(5)`

Ce format est utilisé pour faire référence à une page de manuel spécifique (appelée ci-après simplement page « man »). Le nombre entre parenthèses indique une section spécifique à l'intérieur de **man**. Par exemple, **passwd** a deux pages man. Avec les instructions d'installation de LFS, ces deux pages man seront situées dans `/usr/share/man/man1/passwd.1` et `/usr/share/man/man5/passwd.5`. Ces deux pages renferment 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 correspondant à « passwd » qu'il trouvera, à priori `/usr/share/man/man1/passwd.1`. Dans cet exemple, vous devrez exécuter **man 5 passwd** pour pouvoir lire la page man(5). Notez que la plupart des pages man n'ont pas de noms de page dupliqués dans différentes sections. Du coup, **man *`[nom programme]`*** est généralement suffisant.

6. Structure

Ce livre est divisé selon le découpage suivant :

6.1. Partie I - Introduction

La première partie donne quelques informations importantes sur le déroulement de l'installation 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 dans la construction du système LFS : compilation et installation de tous les paquets un par un, paramétrage des scripts de démarrage et installation du noyau. Le système Linux résultant est la fondation sur 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

Les logiciels utilisés pour créer un système LFS sont constamment mis à jour et améliorés, mais les découvertes de failles de sécurité et les corrections de bogues peuvent survenir après la sortie du livre LFS. Pour vérifier si les paquetages ou instructions de cette version de LFS ont besoin de modifications pour corriger ces bogues, merci de visiter *http://www.linuxfromscratch.org/lfs/errata/6.1.1/* avant de vous lancer. Vous devriez noter toutes les modifications proposées et les appliquer pendant 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, dont 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 hôte pour disposer de ces outils.

Si vous ne désirez pas installer 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, les correctifs et une copie de ce livre. Une fois que vous avez le CD, plus aucune connexion réseau n'est donc 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 l'endroit où le nouveau système LFS sera compilé et installé. Le Chapitre 3 explique quels paquets et correctifs ont besoin d'être téléchargés pour construire un système LFS et comment les stocker sur le nouveau système de fichiers. Le Chapitre 4 traite de la configuration d'un environnement de travail approprié. Merci de lire le Chapitre 4 avec attention, car il présente plusieurs problèmes importants dont le développeur doit être au courant avant de commencer à s'attaquer au Chapitre 5 et aux chapitres suivants.

Le Chapitre 5 détaille l'installation d'un ensemble de paquets qui formeront la suite de développement de base (ou ensemble d'outils) utilisée pour construire le système final 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 également 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 seconde fois). La prochaine étape consiste à construire Glibc, la bibliothèque C, qui sera compilée à l'aide des outils construits lors de la première passe. Ensuite, une seconde passe de l'ensemble d'outils sera construite. Cette fois, l'ensemble d'outils sera lié dynamiquement à la Glibc nouvellement construite. Les paquets restants du Chapitre 5 seront construits en utilisant la seconde passe de l'ensemble d'outils. Lorsque ceci sera fait, le processus d'installation de LFS sera indépendant de la distribution hôte, à l'exception du noyau en cours d'exécution.

Cet effort pour 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 la partition LFS. Ceci s'apparente à redémarrer l'ordinateur et à demander au noyau de monter la partition LFS en tant que partition racine. Le système ne redémarre pas réellement mais change la racine, car la création d'un système amorçable réclame un travail supplémentaire qui n'est pas encore nécessaire. Le principal avantage 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 utiliser son ordinateur comme d'habitude.

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

suite de l'expérience LFS après ce livre. Après avoir accompli toutes les étapes de ce livre, l'ordinateur sera prêt à redémarrer sous le nouveau système LFS.

Voilà pour un court résumé du processus. Des informations détaillées sur chaque étape sont présentées dans les chapitres suivants, avec les descriptions des paquets. Les éléments qui peuvent sembler compliqués seront clarifiés et tout se mettra en place au fur et à mesure de 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 vers ion 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 une indication queun 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 McMurchy 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 rtld, 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 modification 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 problèmes lors de la construction du système LFS, si vous avez des questions ou si vous pensez qu'il y a une coquille typographique 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, l'étape suivante est de chercher dans les liste 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*.

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). Vous trouverez le réseau IRC à l'adresse `irc.linuxfromscratch.org`, port 6667. Le canal dédié au support se nomme #LFS-support.

1.3.5. Références

Pour plus d'informations sur les paquets, des indications utiles sont disponibles sur 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 tout autour du monde pour faciliter l'accès au site web ainsi que le téléchargement des paquetages requis. Merci de visiter le site web de LFS sur *http://www.linuxfromscratch.org/mirrors.html* pour obtenir une liste des miroirs à jour.

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, consultez la FAQ, située sur la page *http://www.linuxfromscratch.org/faq/#generalfaq*. Beaucoup de questions y trouvent leurs réponses. Si ce n'est pas le cas, 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-le dans les listes de discussion sur *http://www.linuxfromscratch.org/search.html*.

Nous avons aussi une formidable communauté LFS, disposée à offrir une assistance via les listes de discussion et IRC (voir la section Section 1.3, « Ressources » de ce livre). Néanmoins, nous recevons de nombreuses questions de support chaque jour, et un grand nombre d'entre elles ont déjà 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 problèmes inhabituels. Si vos recherches sont infructueuses, merci d'inclure toutes informations adéquates (mentionnées ci-dessous) dans votre demande d'assistance.

1.4.1. Éléments à mentionner

En plus d'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
- Signalez si vous avez dévié un tant soit peu des intructions du livre

Note

Dévier du livre ne signifie *pas* que nous n'allons pas vous aider. Après tout, LFS est une histoire de préférences de chaque 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**, examinez le fichier `config.log`. Ce fichier pourrait contenir des 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

La sortie écran et le contenu de différents fichiers sont utiles pour déterminer la cause des problèmes de compilation. Les sorties écran des scripts **configure** et **make** peuvent être utiles. Il n'est pas nécessaire d'inclure la sortie complète, mais incluez suffisamment d'informations intéressantes. Voici ci-dessous un exemple du type d'informations à inclure de la sortie 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 elle signale seulement que quelque chose s'est mal passé, par *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 sur une partition dédiée. Pour la construction d'un système LFS, nous vous recommandons 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) peut aussi être installé 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* explique comment implémenter ceci, alors que ce livre ne décrit que l'installation sur une partition vierge.

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 doit être votre système Linux principal, d'autres logiciels seront probablement installés et réclameront de l'espace disque supplémentaire (estimation très grossière : 2 à 3 Go). Le système LFS en lui-même n'utilise pas tout cet espace, une grande partie est requise pour fournir un espace temporaire. Compiler des paquets peut en effet demander momentanément 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 d'échange du système LFS peut être la même que celle du système hôte, donc inutile d'en créer une nouvelle si votre système hôte en possède déjà une.

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

Souvenez-vous de la désignation de la nouvelle partition (c'est-à-dire `hda5`). Ce livre y fera référence en tant que partition LFS. Souvenez-vous aussi de la désignation de la partition swap. Ces noms seront nécessaires plus tard pour créer le fichier `/etc/fstab`.

2.3. Créer un système de fichiers sur la partition

Maintenant qu'une partition vierge est prête, le système de fichiers peut être créé. Le système le plus communément utilisé dans le monde Linux est le système de fichiers étendu, deuxième version, (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. Des instructions pour l'installation d'autres systèmes de fichiers sont disponibles sur *http://www.linuxfromscratch.org/blfs/view/svn/ postlfs/filesystems.html*.

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

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

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

Note

Quelques distributions hôtes utilisent un outil de création de systèmes de fichiers (e2fsprogs) offrant des fonctionnalités spécifiques. Ceci peut poser des problèmes lors du démarrage sur votre nouveau LFS au chapitre 9, car toutes ces fonctionnalités ne seront pas supportées par l'e2fsprogs installé par LFS vous obtiendriez une erreur semblable à « unsupported filesystem features, upgrade your e2fsprogs ». Pour vérifier si votre système hôte utilise des fonctionnalités spécifiques, 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 fonctionnalités spécifiques. Dans ce cas, pour éviter tout problème ultérieur, vous devez compiler le paquetage e2fsprogs standard et utiliser les binaires résultant de cette compilation pour re-créer 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 avant de pouvoir être utilisée, en exécutant la commande ci-dessous. Si vous utilisez déjà une partition de swap, il n'est pas nécessaire de la reformater.

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

Remplacez *[yyy]* par le nom de la partition de swap.

2.4. Monter la nouvelle partition

Maintenant qu'un système de fichiers a été créé, la partition doit être accessible. Pour cela, la partition a besoin d'être montée sur un point de montage choisi. Dans 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 entrant :

```
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]`* par le nom de la partition LFS.

Si vous utilisez plusieurs partitions pour LFS (c'est-à-dire 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]`* par les noms de partition appropriés.

Assurez-vous que cette nouvelle partition n'est pas montée avec des droits trop restrictifs (telles 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 sans ces options.

Maintenant que nous avons établi un endroit pour travailler, il est temps de télécharger ces paquets.

Chapitre 3. Paquets et correctifs

3.1. Introduction

Ce chapitre contient une liste de paquets à télécharger pour construire un système Linux basique. Les versions indiquées sont celles dont le bon fonctionnement à été vérifié, et le livre suppose leur utilisation. Nous vous recommandons fortement de ne pas utiliser de versions plus récentes, car les commandes de compilation pourraient ne plus être adaptées, et ces versions pourraient poser des problèmes nécessitant des solutions spécifiques. Ces solutions sont développées et stabilisées dans la version de développement du livre.

Les sites de téléchargements pourraient ne pas être toujours accessibles. Si un site a changé depuis la publication de ce livre, Google (*http://www.google.com/*) se révèle être un outil de recherche puissant pour la plupart des paquets. Si cette recherche n'a pas de succès, essayez un autre moyen de téléchargement, dont il est question sur *http://www.linuxfromscratch.org/ lfs/packages.html*.

Les paquets et les correctifs téléchargés devraient être stockés dans un endroit facilement accessible durant toute la construction. Un répertoire de travail est aussi requis pour déballer les sources et pour la construction. Le répertoire `$LFS/sources` peut être utilisé à la fois comme emplacement de stockage pour les archives tar et les correctifs, mais également comme répertoire de travail. En utilisant ce répertoire, les éléments requis seront situés sur la partition LFS, et donc disponibles à toutes les étapes du processus de construction.

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

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

Rendez ce répertoire sticky et autorisez l'écriture. « Sticky » signifie que même si plusieurs utilisateurs peuvent écrire dans un répertoire, seul le propriétaire d'un fichier peut le supprimer. La commande ci-dessous active 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, susceptible de disposer de la bonne version, est :*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.sethwklein.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/
- Mktemp (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, susceptible de disposer de 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. Ils corrigent certaines erreurs contenues dans les paquets, qui seront plus tard éradiquées par les mainteneurs. Certains apportent 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
- 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
- Bzip2 Bzgrep Security Fixes Patch - 1 Ko :
  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
- 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
- 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
- GCC Specs Patch - 14 Ko :
  http://www.linuxfromscratch.org/patches/lfs/6.1.1/gcc-3.4.3-specs-2.patch
- Glibc Rtld Search Dirs Patch - 1 Ko :
  http://www.linuxfromscratch.org/patches/lfs/6.1.1/glibc-2.3.4-rtld_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
- Glibc TLS Assertion Patch - 6 Ko :
  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
- Inetutils Kernel Headers Patch - 1 Ko :
  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
- Mktemp Tempfile Patch - 3 Ko :
  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
- Texinfo Tempfile Fix Patch - 2 Ko :
  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
- Vim Security Patch - 8 Ko :
  http://www.linuxfromscratch.org/patches/lfs/6.1.1/vim-6.3-security_fix-2.patch

En plus des correctifs requis ci-dessus, il en existe un certain nombre d'optionnels, créés par la communauté LFS. Ceux-ci résolvent des problèmes mineurs ou activent des fonctionnalités qui ne sont pas disponibles par défaut. Vous pouvez à loisir consulter la base de données des correctifs sur *http://www.linuxfromscratch.org/patches/* et récupérer ceux correspondants 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 vitale 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 le résultat soit bien 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 ce n'est pas le cas, vous pouvez initialiser la variable avec :

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

Avoir cette variable initialisée est intéressant car des commandes telles que **mkdir $LFS/tools** peuvent être saisies litéralement. Votre shell remplacera « $LFS » par « /mnt/lfs » (ou par ce que vous avez placez dans la variable) lorsqu'il exécutera la ligne de commande.

N'oubliez pas de vérifier que $LFS est initialisé à chaque fois que vous ré-entrez dans l'environnement de travail (par exemple, lorsque vous faites un « su » vers *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 séparer des programmes compilés dans le Chapitre 6. Les programmes compilés ici ne sont que des outils temporaires et ne prendront pas part au système LFS final. En les plaçant dans un répertoire distinct, nous pourrons facilement les supprimer plus tard. Ceci évite également qu'ils ne s'installent dans l'arborescence de l'hôte (facile à faire par accident dans le Chapitre 5).

Créez ce répertoire 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. Exécutez 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 donc 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 toujours 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*, une seule erreur peut endommager, voire dévaster votre système. Du coup, nous vous recommandons de construire les paquets de ce chapitre en tant qu'utilisateur non privilégié. Vous pouvez bien sûr utiliser votre propre nom d'utilisateur, mais pour vous assurer un environnement de travail propre, il est plus sûr de créer un nouvel utilisateur nommé *lfs* comme membre d'un nouveau groupe *lfs* et de l'utiliser lors du processus d'installation. En tant que *root*, exécutez les commandes suivantes pour créer le nouvel utilisateur :

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

Voici la signification des options de la ligne de commande :

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

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

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

`-k /dev/null`
Ce paramètre empêche la copie de fichiers provenant du répertoire squelette (par défaut, `/etc/skel`) en le remplaçant 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 que *lfs* ait un mot de passe), donnez un mot de passe à *lfs* :

```
passwd lfs
```

Donnez-lui un accès complet à `$LFS/tools` en le désignant comme propriétaire du répertoire :

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

Si vous avez créé un répertoire de travail séparé comme suggéré, faites de même avec ce répertoire :

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

Ensuite, connectez-vous en tant que *lfs*. Vous pouvez le faire via une console virtuelle, avec un 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 est décrite dans la page man `bash(1)`, et par **info bash**.

4.4. Configurer l'environnement

Définissez 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 paramètrages et variables d'environnement), puis `.bash_profile`. La commande **exec env -i.../bin/bash** dans le fichier `.bash_profile` remplace le shell en cours par un nouveau avec un environnement entièrement vide, à l'exception des variables `HOME`, `TERM` et `PS1`. Cette technique nous assure un environnement prore en empêchant les variables d'environnement du système hôte de s'inviter dans le système LFS, au risque d'y faire des dégats.

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 que **bash** utilise pour se souvenir du chemin d'accès aux fichiers exécutables, évitant ainsi de reparcourir le `PATH` quand un exécutable est lancé une seconde fois. Cependant, nous voulons utiliser les nouveaux outils dès leur installation. En désactivant la fonction de hachage, le shell parcourt le `PATH` à chaque appel d'un programme. Ainsi, le shell utilisera les nouveaux outils présents dans `$LFS/tools` au fur et à mesure de leur apparition, au lieu de réutiliser sans cesse les versions originales de ces programmes dont la fonction de hachage garderait la trace.

Attribuer la valeur 022 au masque de création de fichier (umask) nous assure que les nouveaux fichiers et répertoires créés ne seront modifiables que 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 les droits 755).

La variable `LFS` devrait pointer sur le point de montage choisi.

La variable `LC_ALL` contrôle la localisation de certains programmes, adaptant leurs messages aux conventions du pays spécifié. Si le système hôte utilise une version de Glibc plus ancienne que la 2.2.4, avoir `LC_ALL` initialisée 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 prévu dans l'environnement chroot.

En plaçant `/tools/bin` au début du `PATH` standard, tous les programmes installées dans Chapitre 5 sont utilisés 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 terminer la préparation de l'environnement en vue de la construction des outils temporaires, chargez le profil de l'utilisateur tout juste créé :

```
source ~/.bash_profile
```

4.5. À propos des SBU

Beaucoup de personnes souhaitent savoir combien de temps la compilation et l'installation de chaque paquet va prendre. Mais, Linux from Scratch est construit sur tant de systèmes différents qu'il est impossible de donner des temps universels. Le plus gros paquet (Glibc) prendra approximativement vingt minutes sur les systèmes les plus rapides mais pourrait prendre jusqu'à trois jours sur les moins rapides. Au lieu d'indiquer des temps en minutes, on les donnera en nombre d'unités de standard de construction (*Standard Build Unit*).

La mesure SBU fonctionne ainsi. Le premier paquet que vous compilerez dans ce livre est Binutils, dans le Chapitre 5. Le temps que prend la compilation de ce paquet est ce que nous appelerons « SBU ». Tous les autres temps de compilation seront exprimés relativement à ce temps.

Par exemple, considérez un paquet dont le temps de compilation est de 4,5 SBU. Ceci signifie que s'il vous a fallu 10 minutes pour compiler et installer la première passe de Binutils, alors vous aurez besoin d' 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 que vous employez. Notez que les SBU sont encore moins précis sur les machines multi-processeurs (SMP). Ils ne sont qu'une estimation du temps nécessaire à l'installation d'un paquetage, et la durée effective peut varier de plusieurs dizaines de minutes dans certains cas.

Si vous souhaitez voir des durées réelles 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. Exécuter cette suite de tests pour un paquet nouvellement construit est généralement une bonne idée car cela confirme que tout a été compilé correctement. Une suite de tests réussissant l'ensemble des vérifications prouve généralement que le paquet fonctionne comme le développeur l'a désiré. Néanmoins, cela ne garantit pas que le paquet ne contient pas de bogues.

Certaines suites de tests sont plus importantes que d'autres. Par exemple, les suites de tests des paquets formant le cœur de la chaîne de compilation (GCC, Binutils, et Glibc -la bibliothèque C-) sont de la plus haute 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 elles 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'est pas possible de se soustraire à l'influence exercée par le système hôte lors des tests de ce chapitre, occasionnant fréquemment des échecs étonnants et inexplicables. Comme les outils construits dans le Chapitre 5 sont temporaires et ultérieurement supprimés, nous recommandons à l'utilisateur standard de ne pas lancer les suites de tests dans ce chapître. Les instructions pour lancer ces suites de test sont fournies pour les testeurs et les développeurs mais elles sont réellement optionnelles pour toutes les autres personnes.

Un problème courant lors de l'exécution des suites de test de Binutils et GCC est de manquer de pseudo-terminaux (PTY). Le symptôme est un nombre élevé de tests ayant échoués. Plusieurs causes sont possibles, mais la plus raisonnable est que le système hôte ne dispose pas d'un système de fichiers `devpts` configuré correctement. Ce problème est discuté plus en détails dans le Chapitre 5.

Quelques fois, les suites de test des paquets échoueront pour des raisons dont les développeurs sont conscients mais qu'ils n'ont pas estimées critiques. Consultez les traces situées sur *http://www.linuxfromscratch.org/lfs/build-logs/6.1.1/* pour vérifier si ces échecs sont attendus. Ce site s'applique à 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 produit 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 triplette cible. De nombreuses fois, la triplette cible sera probablement *i686-pc-linux-gnu*. Une façon simple de déterminer le nom de la triplette cible est de lancer le script **config.guess** venant avec le source pour un grand nombre de 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 lancé 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 expliquer 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

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

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 garantie 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 avions 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**) rejetteront 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 recommander 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 contruira 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 modifé 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 regénérée :

```
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-unneeded /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-unneeded`* 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* *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*.

Note

Le reste de ce libre 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 dûe 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—ils 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 rappelerait 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 créé 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 configure :

`--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 laissez 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 paramétrages 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 dex 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 officieux 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 installés 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

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

mtrace	Lit et interprète un fichier de trace mémoire et affiche un résumé dans un format lisible par un humain
nscd	Un démon pour les services de noms fournissant un cache pour les requêtes les plus communes
nscd_nischeck	Vérifie si le mode sécurisé est nécessaire pour les recherches NIS+
pcprofiledump	Affiche des informations générées par un profilage du PC
pt_chown	un programme d'aide pour que **grantpt** initialise les droits des propriétaires, groupes et autres d'un pseudo-terminal esclave
rpcgen	Génère du code C pour implémenter le protocole RPC (*Remote Procecure Call*)
rpcinfo	Fait un appel RPC à un serveur RPC
sln	Un programme **ln** lié statiquement
sprof	Lit et affiche les données de profilage des objets partagés
tzselect	Demande à l'utilisateur l'emplacement géographique du système et donne la description du fuseau horaire correspondante
xtrace	Trace l'exécution d'un programme en affichant la fonction en cours d'exécution
zdump	Afficheur de fuseau horaire
zic	Compilateur de fuseau horaire
`ld.so`	Le programme d'aide des bibliothèques partagées exécutables
`libBrokenLocale`	Utilisé 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 (*Institute of Electrical and Electronic Engineers*)
`libm`	La bibliothèque mathématique
`libmcheck`	Contient du code à lancer au démarrage
`libmemusage`	Utilisé par **memusage** 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

`libpcprofile`	Contient des fonctions de profilage utilisées pour tracer le temps CPU dépensé sur les lignes de code source
`libpthread`	La bibliothèque threads POSIX
`libresolv`	Contient des fonctions de création, d'envoi et d'interprétation de paquets pour les serveurs de noms de domaine Internet
`librpcsvc`	Contient des fonctions apportant différents services RPC
`librt`	Contient des fonctions fournissant la plupart des interfaces spécifiées par l'extension temps réel de POSIX.1b
`libthread_db`	Contient des fonctions utiles pour construire des débogueurs de programmes multi-threads
`libutil`	Contient du code pour les fonctions « standard » utilisées par de nombreux outils Unix

6.12. 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 le s 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 exécutables) 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

addr2line Traduit les adresses de programme en noms de fichier et numéros de ligne ; suivant une adresse et le nom d'un exécutable, il utilise les informations de débogage disponibles dans l'exécutable pour déterminer le fichier source et le numéro de ligne associé à cette adresse

ar Crée, modifie et extrait à partir d'archives

as Un assembleur qui assemble la sortie de **gcc** en un fichier objet

c++filt Utilisé par l'éditeur de liens pour récupérer les symboles C++ et Java, et pour empêcher les fonctions surchargées d'arrêter brutalement le programme

gprof Affiche les données de profilage d'appels dans un graphe

ld Un éditeur de liens combinant un certain nombre d'objets et de fichiers archives en un seul fichier, en déplaçant leur données et en regroupant les références de symboles

nm Liste les symboles disponibles dans un fichier objet

objcopy Traduit un type de fichier objet en un autre

objdump Affiche des informations sur le fichier objet donné, les options contrôlant les informations à afficher ; l'information affichée est surtout utile aux programmeurs qui travaillent sur les outils de compilation

ranlib Génère un index du contenu d'une archive et le stocke dans l'archive ; l'index liste tous les symboles définis par les membres de l'archive qui sont des fichiers objet déplaçables

readelf Affiche des informations sur les binaires du type ELF

size Liste les tailles des sections et la taille totale pour les fichiers objets donnés

strings Affiche, pour chaque fichier donné, la séquence de caractères affichables qui sont d'au moins la taille spécifiée (par défaut, 4) ; pour les fichiers objets, il affiche, par défaut, seulement les chaînes des sections d'initialisation et de chargement alors que pour les

	autres types de fichiers, il parcourt le fichier entier
strip	Supprime les symboles des fichiers objets
`libiberty`	contient des routines utilisées par différents programmes GNU comme **getopt**, **obstack**, **strerror**, **strtol** et **strtoul**
`libbfd`	La bibliothèque des descripteurs de fichiers binaires (*Binary File Descriptor*)
`libopcodes`	Une bibliothèque de gestion des opcodes—la « version lisible » des instructions du processeur ; elle est utilisée pour construire des outils comme **objdump**.

6.14. 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,so] et libsupc++.a

Descriptions courtes

cc	Le compilateur C
cpp	Le préprocesseur C ; il est utilisé par le compilateur pour l'extension des instructions #include, #define et d'autres instructions similaires dans les fichiers sources
c++	Le compilateur C++
g++	Le compilateur C++
gcc	Le compilateur C
gccbug	Un script shell utilisé pour aider à la création de bons rapports de bogues

gcov	Un outil de tests ; il est utilisé pour analyser les programmes et savoir où des optimisations seraient suivies du plus d'effet
`libgcc`	contient un support en exécution pour **gcc**
`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

basename Supprime tout le chemin et un suffixe donné à partir du nom de fichier donné

cat Concatène des fichiers sur la sortie standard

chgrp Change le groupe propriétaire de certains fichiers et répertoires.

chmod Change les droits de chaque fichier donné avec le mode indiqué. Le mode peut être soit une représentation symbolique des modifications à faire soit un nombre octal représentant les nouveaux droits

chown Modifie le propriétaire utilisateur et/ou groupe de certains fichiers et répertoires

chroot Lance une commande avec le répertoire spécifié comme répertoire racine (`/`)

cksum Affiche la somme de vérification CRC (Cyclic Redundancy Check) et le nombre d'octets de chaque fichier

comm Compare deux fichiers triés, affichant sur trois colonnes, les lignes uniques et les lignes communes

cp Copie des fichiers

csplit Divise un fichier donné sur plusieurs fichiers indiqués, les séparant par des modèles donnés ou des numéros de lignes. Il affiche le nombre total d'octets pour chaque nouveau fichier

cut Affiche des parties de lignes, sélectionnant ces parties suivant des champs ou positions donnés

date Affiche l'heure actuelle dans le format donné ou initialise la date système

dd Copie un fichier en utilisant la taille de bloc donnée et le nombre, tout en réalisant des conversions optionnelles

df Affiche l'espace disque disponible (et utilisé) sur tous les systèmes de fichiers montés, ou seulement sur les systèmes de fichiers contenant les fichiers donnés

dir Liste le contenu de chaque répertoire donné (identique à la commande **ls**)

dircolors Affiche les commandes pour initialiser la variable d'environnement `LS_COLOR`, ce qui permet de changer le schéma de couleurs utilisé par **ls**

dirname Supprime le suffixe qui ne représente pas le répertoire dans un nom de fichier donné

du Affiche le total de l'espace disque utilisé par le répertoire actuel, ou par chacun des répertoires donnés incluant tous les sous-répertoires, ou par chacun des fichiers donnés

echo Affiche les chaînes données

env Lance une commande dans un environnement modifié

expand Convertit les tabulations en espaces

expr Évalue des expressions

factor Affiche les facteurs premiers de tous les entiers spécifiés

false Ne fait rien. Il renvoie toujours un un code d'erreur indiquant l'échec

fmt Reformate les paragraphes dans les fichiers donnés

fold Emballe les lignes des fichiers donnés

groups Affiche les groupes auxquels appartient un utilisateur

head Affiche les dix premières lignes (ou le nombre demandé de lignes) pour chaque fichier précisé

hostid Affiche l'identifieur numérique de l'hôte en hexadécimal

hostname Affiche ou initialise le nom de l'hôte

id Affiche l'identifieur effectif de l'utilisateur courant ou de l'utilisateur précisé, l'identifieur du groupe et les groupes auxquels appartient cet utilisateur

install Copie les fichiers en initialisant leur droits et, si possible, leur propriétaire et groupe

join Joint à partir de deux fichiers les lignes qui ont des champs de jointure identiques

link Crée un lien physique avec le nom de donné vers le fichier indiqué

ln Crée des liens symboliques ou physiques entre des fichiers

logname Indique le nom de connexion de l'utilisateur actuel

ls Liste le contenu de chaque répertoire donné

md5sum Affiche ou vérifie les sommes de vérification MD5 (Message Digest 5)

mkdir Crée des répertoires avec les noms donnés

mkfifo Crée des fichiers FIFOs (First-In, First-Out, un « tube nommé » dans le vocable d'Unix) avec les noms donnés

mknod Crée des noeuds périphérique avec les noms donnés. Un noeud périphérique est un fichier spécial de type caractère ou bloc, ou encore un FIFO

mv Déplace ou renomme des fichiers ou répertoires

nice Lance un programme avec un priorité modifiée

nl Numérote les lignes des fichiers donnés

nohup Lance une commande immune aux arrêts brutaux, dont la sortie est redirigée vers un journal de traces

od Affiche les fichiers en octal ou sous d'autres formes

paste Joint les fichiers donnés en faisant plaçant les lignes correspondantes ligne par ligne en les séparant par des caractères de tabulation

pathchk Vérifie que les noms de fichier sont valides ou portables

pinky Un client « finger » léger. Il affiche quelques informations sur les utilisateurs indiqués

pr Fait de la pagination, principalement en colonne, des fichiers pour une impression

printenv Affiche l'environnement

printf Affiche les arguments donnés suivant le format demandé, un peu comme la fonction C printf

ptx Produit un index permuté à partir du contenu des fichiers indiqués, avec le contexte de chaque mot

pwd Indique le nom du répertoire courant

readlink Indique la valeur du lien symbolique

rm Supprime des fichiers ou des répertoires

rmdir Supprime des répertoires s'ils sont vides

seq Affiche une séquence de nombres, à l'intérieur d'une échelle et avec un incrément spécifié

sha1sum Affiche ou vérifie des sommes de vérification 160-bit SHA1

shred Efface les fichiers indiqués en écrivant dessus des modèles aléatoires pour rendre la récupération des données très difficile

sleep Fait une pause d'un certain temps

sort Trie les lignes des fichiers donnés

split Divise les fichiers donnés en plusieurs pièces, par taille ou par nombre de lignes

stat Affiche le statut du fichier ou du système de fichiers

stty Initialise ou affiche les paramètres de la ligne du terminal

sum Affiche la somme de vérification et le nombre de blocs pour chacun des fichiers donnés

sync Vide les tampons du système de fichiers. Cela force l'enregistrement des blocs modifiés sur disque et met à jour le superbloc

tac Concatène les fichiers donnés à l'envers

tail Affiche les dix dernières lignes (ou le nombre de lignes indiqué) pour chaque fichier précisé

tee Lit à partir de l'entrée standard en écrivant à la fois sur la sortie standard et sur les fichiers indiqués

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

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

`/etc/protocols`	Décrit les différents protocoles Internet DARPA disponibles à partir du sous-système TCP/IP
`/etc/services`	Fournit une correspondance entre des noms de services internet et leur numéros de port et types de protocoles affectés

6.19. 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

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

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

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

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

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

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 inexistants. 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: captoinfo (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

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

6.22. 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 multioctets. Ceci est nécessaire dans le cas d'une utilisation d'une locale avec un ensemble de caractères multi-octets. Ce commutateur peut aussi être utile pour avoir la capacité d'éditer des fichiers créés initialement avec des distributions Linux comme Fedora 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

efm_filter.pl	Un filtre pour créer un fichier d'erreur pouvant être lu par **vim**
efm_perl.pl	Reformate les messages d'erreur de l'interpréteur Perl pour utiliser le mode « quickfix » de **vim**

ex	Lance **vim** en mode ex
less.sh	Un script qui exécute **vim** avec less.vim
mve.awk	Traite les erreurs de **vim**
pltags.pl	Crée un fichier de balises pour le code Perl, utilisé par **vim**
ref	Vérifie la validité des arguments
rview	Une version restreinte de **view** : aucune commande shell ne peut être lancée et **view** ne peut pas être suspendu
rvim	Une version restreinte de **vim** : aucune commande shell ne peut être lancée et **vim** ne peut pas être suspendu
shtags.pl	Génère un fichier de balises pour les scripts Perl
tcltags	Génère un fichier de balises pour le code TCL
view	Lance **vim** en mode lecture seule
vi	L'éditeur
vim	L'éditeur
vim132	Lance **vim** avec le terminal en mode 132 colonnes
vim2html.pl	Convertit la documentation de Vim en HTML
vimdiff	Édite deux ou trois versions d'un fichier avec **vim** et montre les différences
vimm	Active le modèle d'entrée DEC sur un terminal distant
vimspell.sh	Un script qui vérifie un fichier et génère les instructions de syntaxe nécessaires pour mettre en surlignage dans **vim**. Ce script requiert l'ancienne commande Unix **spell**, qui n'est fourni ni par LFS ni par BLFS
vimtutor	Vous apprend les touches et les commandes basiques de **vim**
xxd	Fait un affichage hexa du fichier donné. Il peut aussi faire l'inverse pour une correspondance binaire

6.24. 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

less un visualisateur de fichiers. Il affiche le contenu du fichier donné, vous permettant d'aller vers le haut et vers le bas, de chercher des chaînes et de sauter vers des repères

lessecho nécessaire pour étendre les meta-caractères, comme * et ?, dans les noms de fichiers de systèmes Unix

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

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

addftinfo	Lit un fichier de polices troff et ajoute quelques informations métriques supplémentaire sur la police qui est utilisé par le système **groff**
afmtodit	Crée un fichier de police à utiliser avec **groff** et **grops**
geqn	Compile les descriptions d'équations imbriquées dans les fichiers d'entrées de troff pour obtenir des commandes comprises par **troff**
eqn2graph	Convertit une équation EQN troff en une image améliorée
eqn	Un lien vers **eqn**
grn	Un préprocesseur **groff** pour les fichiers gremlin

grodvi Un pilote pour **groff** qui produit un format dvi TeX

groff Une interface au système de formatage de document groff. Normalement, il lance le programme **troff** et un post-processeur approprié au périphérique sélectionné

groffer Affiche des fichiers groff et des pages man sur des terminaux X et tty

grog Lit des fichiers et devine les options *-e*, *-man*, *-me*, *-mm*, *-ms*, *-p*, *-s* et *-t* de **groff**, requises pour l'impression des fichiers. Il indique la commande **groff** incluant ces options

grolbp Un pilote **groff** pour les imprimantes Canon CAPSL (imprimantes laser de la série LBP-4 et LBP-8)

grolj4 Un pilote pour **groff** produisant une sortie au format PCL5, intéressant les imprimantes HP Laserjet 4

grops Traduit la sortie de GNU **troff** en Postscript

grotty Traduit la sortie de GNU **troff** en un format compatible pour les périphériques de type machine à écrire

gtbl Un lien vers **tbl**

hpftodit Crée un fichier de polices à utiliser avec **groff -Tlj4** à partir d'un fichier métrique de police HP

indxbib Crée un index inversé d'un fichier spécifié, index utilisé par les bases de données bibliographiques avec **refer**, **lookbib** et **lkbib**

lkbib recherche dans les bases de données bibliographiques pour des références contenant certaines clés et indique toute référence trouvée

lookbib Affiche une invite sur la sortie des erreurs (sauf si l'entrée standard n'est pas un terminal), lit à partir de l'entrée standard une ligne contenant un ensemble de mots clés, recherche dans les bases de données bibliographiques dans un fichier spécifié les références contenant ces mots clés, affiche toute référence trouvée sur la sortie standard et répère ce processus jusqu'à la fin de l'entrée

mmroff Un pré-processeur pour **groff**

neqn Formate les équations pour une sortie ASCII (*American Standard Code for Information Interchange*)

nroff Un script qui émule la commande **nroff** en utilisant **groff**

pfbtops Traduit une police Postscript au format `.pfb` en ASCII

pic Compile les descriptions d'images embarquées à l'intérieur de fichiers d'entrées troff ou TeX en des commandes comprises par TeX ou **troff**

pic2graph Convertit un diagramme PIC en une image améliorée

post-grohtml Traduit la sortie de GNU troff **troff** en html

pre-grohtml Traduit la sortie de GNU **troff** en html

refer Copie le contenu d'un fichier sur la sortie standard, sauf pour les lignes entre les symboles *.[* et *.]* interprétées comme des citations, et les lignes entre *.R1* et *.R2* interprétées comme des commandes sur la façon de gérer les citations

soelim Lit des fichiers et remplace les lignes de la forme *.so fichier* par le contenu du *fichier*

	mentionné
tbl	Compile les descriptions des tables imbriquées dans les fichiers d'entrées troff en commandes comprises par **troff**
tfmtodit	Crée un fichier de police à utiliser avec **groff -Tdvi**
troff	Est hautement compatible avec la commande Unix **troff**. Habituellement, il devrait être appelé en utilisant la commande **groff**, qui lance aussi les pré-processeurs et post-processeurs dans l'ordre approprié et avec les options appropriées
zsoelim	Un lien vers **soelim**

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 regénérée :

```
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

autopoint	Copie les fichiers d'infrastructure standard gettext en un paquet source
config.charset	Affiche une table des caractères dépendante du système
config.rpath	Affiche un ensemble de variables dépendant du système, décrivant comment initialiser le chemin de recherche à l'exécution des bibliothèques partagées dans un exécutable
envsubst	Substitue les variables d'environnement dans des chaînes de format shell
gettext	Traduit un message en langue naturelle dans la langue de l'utilisateur en recherchant la traduction dans un catalogue de messages
gettextize	Copie tous les fichiers standard Gettext dans le répertoire de haut niveau d'un paquet, pour commencer son internationalisation
hostname	Affiche un nom d'hôte réseau sous plusieurs formats

msgattrib	Filtre les messages d'un catalogue de traduction suivant leurs attributs et manipule les attributs
msgcat	Concatène et fusionne les fichiers `.po` donnés
msgcmp	Compare deux fichiers `.po` pour vérifier que les deux contiennent le même ensemble de chaînes msgid
msgcomm	trouve les messages qui sont communs aux fichiers `.po`
msgconv	Convertit un catalogue de traduction en un autre codage de caractères
msgen	Crée un catalogue de traduction anglais
msgexec	Applique une commande pour toutes les traductions d'un catalogue de traduction
msgfilter	Applique un filtre à toutes les traductions d'un catalogue de traductions
msgfmt	Génère un catalogue binaire de messages à partir d'un catalogue de traductions
msggrep	Extrait tous les messages d'un catalogue de traductions correspondant à un modèle donné ou appartenant à d'autres sources données
msginit	Crée un nouveau fichier `.po`, initialisant la méta-information avec des valeurs de l'environnement de l'utilisateur
msgmerge	Combine deux traductions brutes en un seul fichier
msgunfmt	Décompile un catalogue de messages binaire en un texte brut de la traduction
msguniq	Unifie les traductions dupliquées en un catalogue de traduction
ngettext	Affiche les traductions dans la langue native d'un message texte dont la forme grammaticale dépend d'un nombre
xgettext	Extrait les lignes de messages traduisibles à partir des fichiers source donnés pour réaliser la première traduction de modèle
`libasprintf`	Définit la classe *autosprintf*, qui rend les routines de sortie formatée C utilisables dans les programmes C++ pour utiliser les chaînes <*string*> et les flux <*iostream*>
`libgettextlib`	Une bibliothèque privée contenant les routines communes utilisées par les nombreux programmes gettext. Ils ne sont pas fait pour une utilisation générale
`libgettextpo`	Utilisé pour écrire les programmes spécialisés qui s'occupent des fichiers `.po`. Cette bibliothèque est utilisée lorsque les applications standards livrés avec Gettext ne vont pas suffire (comme **msgcomm**, **msgcmp**, **msgattrib** et **msgen**)
`libgettextsrc`	Une bibliothèque privée contenant les routines communes utilisées par les nombreux programmes gettext. Elles ne sont pas destinées à une utilisation générale

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

ftp	Est un programme pour le protocole de transfert de fichier
ping	Envoie des paquets echo-request et affiche le temps mis pour que la réponse arrive
rcp	Fait une copie de fichiers distants
rlogin	Permet une connexion à distance
rsh	Exécute un shell distant
talk	Est utilisé pour discuter avec un autre utilisateur
telnet	Une interface du protocole TELNET
tftp	Un programme de transfert trivial de fichiers

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 lnstat), ifcfg, ifstat, ip, lnstat, nstat, routef, routel, rtacct, rtmon, rtpr, rtstat (lien vers lnstat), ss et tc.

Descriptions courtes

ctstat Outil donnant le statut de la connexion

ifcfg Un emballage en script shell pour la commande **ip**

ifstat Affiche les statistiques des interfaces, incluant le nombre de de paquets émis et transmis par l'interface.

ip L'exécutable principal. Il a plusieurs fonctions :

ip link *`[périphérique]`* autorise les utilisateurs à regarder l'état des périphériques et à faire des changements.

ip addr autorise les utilisateurs à regarder les adresses et leurs propriétés, à ajouter de nouvelles adresses et à supprimer les anciennes.

ip neighbor autorise les utilisateurs à regarder dans les liens des voisins et dans leurs propriétés, à ajouter de nouvelles entrées et à supprimer les anciennes.

ip rule autorise les utilisateurs à regarder les politiques de routage et à les modifier.

ip route autorise les utilisateurs à regarder la table de routage et à modifier les règles de routage.

ip tunnel autorise les utilisateurs à regarder les tunnels IP et leurs propriétés, et à les modifier.

ip maddr autorise les utilisateurs à regarder adresses multicast et leurs propriétés, et à les changer.

ip mroute autorise les utilisateurs à configurer, modifier ou supprimer le routage multicast.

ip monitor autorise les utilisateurs à surveiller en continu l'état des périphériques, des adresses et des routes.

lnstat Fournit les statistiques réseau Linux. C'est un remplacement plus généraliste et plus complet de l'ancien programme **rtstat**

nstat Affiche les statistiques réseau.

routef Un composant de **ip route** pour vider les tables de routage.

routel Un composant de **ip route** pour afficher les tables de routage.

rtacct Affiche le contenu de `/proc/net/rt_acct`

rtmon Outil de surveillance de routes.

rtpr Convertit la sortie de **ip -o** en un format lisibles

rtstat Outil de statut de routes

ss Similaire à la commande **netstat** ; affiche les connexions actives

tc Exécutable de contrôle du trafic ; utile pour l'implémentation de la qualité de service (QOS) et de la classe de service (COS)

tc qdisc autorise les utilisateurs à configurer la discipline de queues

tc class autorise les utilisateurs à configurer les classes suivant la planification de la discipline de queues

tc estimator autorise les utilisateurs à estimer le flux réseau dans un réseau

tc filter autorise les utilisateurs à configurer les filtres de paquets pour QOS/COS

tc policy autorise les utilisateurs à configurer les politiques QOS/COS

6.33. 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

en2cxs	construit une extension Perl pour le module Encode, soit à partir de *Unicode Character Mappings* soit à partir de *Tcl Encoding Files*
find2perl	Traduit les commandes **find** en Perl
h2ph	Convertit les fichiers d'en-têtes C `.h` en fichiers d'en-têtes Perl `.ph`
h2xs	Convertit les fichiers d'en-têtes C `.h` en extensions Perl
libnetcfg	Peut être utilisé pour configurer `libnet`
perl	Combine quelques-unes des meilleures fonctionnalités de C, **sed**, **awk** et **sh** en un langage style couteau suisse
perl5.8.7	Un lien vers **perl**
perlbug	Utilisé pour générer des rapports de bogues sur Perl ou les modules l'accompagnant et pour les envoyer par courrier électronique
perlcc	Génère des exécutables à partir des programmes Perl
perldoc	Affiche une partie de la documentation au format pod, embarquée dans le répertoire d'installation de Perl ou dans un script Perl
perlivp	La procédure de vérification d'installation de Perl (*Perl Installation Verification Procedure*). Il peut être utilisé pour vérifier que Perl et ses bibliothèques ont été installés correctement
piconv	une version Perl du convertisseur de codage des caractères **iconv**
pl2pm	un outil simple pour la conversion des fichiers Perl4 `.pl` en modules Perl5 `.pm`
pod2html	Convertit des fichiers à partir du format pod vers le format HTML
pod2latex	Convertit des fichiers à partir du format pod vers le format LaTeX
pod2man	Convertit des fichiers à partir du format pod vers une entrée formatée *roff
pod2text	Convertit des fichiers à partir du format pod vers du texte ANSI
pod2usage	Affiche les messages d'usage à partir des documents embarqués pod
podchecker	Vérifie la syntaxe du format pod des fichiers de documentation
podselect	Affiche les sections sélectionnées de la documentation pod
psed	Une version Perl de l'éditeur en flux **sed**
pstruct	affiche les structures C générées à partir de **cc -g -S**
s2p	Traduit les scripts **sed** en perl
splain	est utilisé pour forcer la verbosité des messages d'avertissement avec Perl
xsubpp	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

info	Utilisé pour lire des pages info similaires aux pages man mais qui vont souvent plus loin que la simple explication des arguments disponibles. Par exemple, comparez **man bison** et **info bison**.
infokey	Compile un fichier source contenant des personnalisations Info en un format binaire
install-info	Utilisé pour installer les pages info ; il met à jour les entrées dans le fichier index d'**info**
makeinfo	Traduit les sources Texinfo données dans différents autres langages : pages info, texte ou HTML
texi2dvi	Utilisé pour formater le document Texinfo indiqué en un fichier indépendant des périphériques, pouvant être édité
texi2pdf	Utilisé pour formater le document Texinfo indiqué en un fichier PDF
texindex	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

autoconf Produit des scripts shell configurant automatiquement des paquets de code source, permettant ainsi de les adapter à tous les types de systèmes Unix. Les scripts de configuration qu'il produit sont indépendants. Les exécuter ne nécessite pas le programme **autoconf**.

autoheader Un outil pour créer des fichiers modèle d'instructions C *#define* que configure utilise.

autom4te Un emballage pour le processeur de macro M4.

autoreconf Exécute automatiquement **autoconf**, **autoheader**, **aclocal**, **automake**, **gettextize** et **libtoolize** dans le bon ordre pour gagner du temps lorsque des modifications ont eu lieu sur les fichiers modèles d'**autoconf** et d'**automake**.

autoscan Aide à la création de fichiers `configure.in` pour un paquet logiciel. Il examine les fichiers source d'un répertoire et crée un fichier `configure.scan` utilisant `configure.in` comme base pour le paquet

autoupdate Modifie un fichier `configure.in` qui appelle toujours les macros **autoconf** par leurs anciens noms pour qu'il utilise les noms de macros actuels.

ifnames Sert à écrire les fichiers `configure.in` pour un paquet logiciel. Il affiche les identifieurs que le paquet utilise dans des conditions du préprocesseur C. Si un paquet a déjà été initialisé pour avoir une certaine portabilité, ce programme aide à déterminer ce que **configure** doit vérifier. Il peut aussi remplir les blancs dans un fichier `configure.in` généré par **autoscan**.

6.36. 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

acinstall	Un script qui installe des fichiers M4, style aclocal
aclocal	Génère des fichiers `aclocal.m4` basés sur le contenu de fichiers `configure.in`
aclocal-1.9.5	Un lien vers **aclocal**
automake	Un outil pour générer automatiquement des fichiers `Makefile.in` à partir de fichiers `Makefile.am`. Pour créer tous les fichiers `Makefile.in` d'un paquet, lancez ce programme dans le répertoire de haut niveau. En parcourant le fichier `configure.in`, il trouve automatiquement chaque fichier `Makefile.am` approprié et génère le fichier `Makefile.in` correspondant
automake-1.9.5	Un lien vers **automake**
compile	Un emballage pour les compilateurs
config.guess	Un script qui tente de deviner un triplet canonique pour la construction donnée, l'hôte ou l'architecture de la cible
config.sub	Un script contenant une sous-routine de validation de configuration

depcomp	Un script pour compiler un programme de façon à ce que les informations de dépendances soient générées en plus de la sortie désirée
elisp-comp	Compile le code Lisp d'Emacs
install-sh	Un script qui installe un programme, un script ou un fichier de données
mdate-sh	Un script qui affiche la date de modification d'un fichier ou répertoire
missing	Un script agissant comme remplaçant pour les programmes GNU manquants lors d'une installation
mkinstalldirs	Un script qui crée un ensemble de répertoires
py-compile	Compile un programme Python
symlink-tree	Un script créant un ensemble de liens à partir d'un ensemble de répertoires
ylwrap	Un emballage pour **lex** et **yacc**

6.37. 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 indiqué 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

file	Tente de classifier chaque fichier donné. Il réalise ceci en exécutant différents tests : tests sur le système de fichiers, tests des nombres magiques et tests de langages
`libmagic`	Contient des routines pour la reconnaissance de nombres magiques utilisés par le programme **file**

6.39. 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

libtool	Fournit des services de support de construction généralisée de bibliothèques
libtoolize	Fournit une façon standard d'ajouter le support de **libtool** dans un paquet
`libltdl`	Cache les nombreuses difficultés avec dlopen sur les bibliothèques

6.40. 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), bzcat (lien vers bzip2), bzcmp, bzdiff, bzegrep, bzfgrep, bzgrep, bzip2, bzip2recover, bzless et bzmore
Bibliothèques installées: libbz2.[a,so]

Descriptions courtes

bunzip2	Décompresse les fichiers compressés avec bzip
bzcat	Décompresse vers la sortie standard
bzcmp	Lance **cmp** sur des fichiers compressés avec bzip
bzdiff	Lance **diff** sur des fichiers compressés avec bzip
bzgrep	Lance **grep** sur des fichiers compressés avec bzip
bzegrep	Lance **egrep** sur des fichiers compressés avec bzip
bzfgrep	Lance **fgrep** sur des fichiers compressés avec bzip
bzip2	Compresse les fichiers en utilisant l'algorithme de compression de texte par tri de blocs de Burrows-Wheeler avec le codage Huffman ; le taux de compression est meilleur que celui auquel parviennent les outils de compression plus conventionnels utilisant les algorithmes « Lempel-Ziv », comme **gzip**
bzip2recover	Essaie de récupérer des données à partir de fichiers endommagés, compressés avec bzip
bzless	Lance **less** sur des fichiers compressés avec bzip
bzmore	Lance **more** sur des fichiers compressés avec bzip
`libbz2*`	La bibliothèque implémentant la compression de données sans perte par tri de blocs, utilisant l'algorithme de Burrows-Wheeler

6.41. 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, deallocvt, 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

chvt	Change le terminal virtuel en avant plan
deallocvt	Désalloue les terminaux virtuels inutilisés
dumpkeys	Affiche les tables de traduction du clavier
fgconsole	Affiche le numéro du terminal virtuel actif
getkeycodes	Affiche la table de correspondance des « scancode » avec les « keycode »
getunimap	Affiche la table de correspondance unicode vers police en cours d'utilisation
kbd_mode	Affiche ou initialise le mode du clavier
kbdrate	Initialise les taux de répétition et de délai du clavier
loadkeys	Charge les tables de traduction du clavier
loadunimap	Charge la table de correspondance entre unicode et police

mapscrn	un programme obsolète utilisé pour charger une table de correspondance des caractères de sortie définie par l'utilisateur dans le pilote de la console. Ceci est maintenant fait par **setfont**
openvt	Lance un programme sur un nouveau terminal virtuel (VT)
psfaddtable	Un lien vers **psfxtable**
psfgettable	Un lien vers **psfxtable**
psfstriptable	Un lien vers **psfxtable**
psfxtable	Gére les tables de caractères Unicode pour les polices de la console
resizecons	Change l'idée du noyau sur la taille de la console
setfont	Modifie les polices EGA/VGA sur la console
setkeycodes	Charge les entrées de la table de correspondance entre scancode et keycode, utile si vous avez des touches inhabituelles sur votre clavier
setleds	Initialise les drapeaux et LED du clavier
setlogcons	Envoie les messages du noyau sur la console
setmetamode	Définit la gestion des touches meta du clavier
setvesablank	Laisse l'utilisateur configurer la sauvegarde d'écran matérielle (un simple écran vide)
showconsolefont	Affiche la police de l'écran pour la console EGA/VGA
showkey	Affiche les scancodes, keycodes et codes ASCII des touches appuyées sur le clavier
unicode_start	Met le clavier et la console en mode UNICODE. Ne l'utilisez jamais sur LFS car les applications ne sont pas configurées pour supporter UNICODE.
unicode_stop	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

badblocks	Recherche les blocs défectueux sur un périphérique (habituellement une partition d'un disque)
blkid	Un outil en ligne de commande pour trouver et afficher les attributs d'un périphérique bloc
chattr	Modifie les attributs de fichiers sur un système de fichiers `ext2` et `ext3`, la version journalisée d'`ext2`
compile_et	Un compilateur de table d'erreurs. Il convertit une table de noms d'erreurs et des messages associés en un fichier source C à utiliser avec la bibliothèque `com_err` library
debugfs	Un débogueur de système de fichiers. Il est utilisé pour examiner et modifier l'état d'un système de fichiers `ext2`
dumpe2fs	Affiche le superbloc et les informations de groupes de blocs sur le système de fichiers présent sur un périphérique donné
e2fsck	Est utilisé pour vérifier, et quelque fois réparé, les systèmes de fichiers `ext2` et `ext3`
e2image	Est utilisé pour sauver les données critiques d'un système de fichiers `ext2` dans un fichier
e2label	Affiche ou modifie le label d'un système de fichiers `ext2` présent sur un périphérique donné
findfs	Trouve un système de fichiers par label ou UUID (*Universally Unique Identifier*, soit Identifiant Unique Universel)
fsck	Est utilisé pour vérifier, et parfois réparer, les systèmes de fichiers
fsck.ext2	Vérifie par défaut les systèmes de fichiers `ext2`
fsck.ext3	Vérifie par défaut les systèmes de fichiers `ext3`
logsave	Sauvegarde la sortie d'une commande dans un journal de traces
lsattr	Liste les attributs de fichiers sur un système de fichiers ext2
mk_cmds	Convertit une table de noms de commandes et de messages d'aide en un fichier source

	C bon à utiliser avec la bibliothèque sous-système `libss`
mke2fs	Crée un système de fichiers ext2 ou ext3 sur le périphérique donné
mkfs.ext2	Crée par défaut un système de fichiers `ext2`
mkfs.ext3	Crée par défaut un système de fichiers `ext3`
mklost+found	Est utilisé pour créer un répertoire `lost+found` sur un système de fichiers `ext2`. Il pré-alloue des blocs disque dans ce répertoire pour faciliter la tâche d'**e2fsck**
resize2fs	est utilisé pour agrandir ou réduire un système de fichiers `ext2`
tune2fs	Est utilisé pour ajuster les paramètres d'un système de fichiers à condition qu'il soit un `ext2`
uuidgen	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.
`libblkid`	Contient des routines pour l'identification de processus et l'extraction de modèles
`libcom_err`	La routine d'affichage d'erreurs
`libe2p`	Est utilisé par **dumpe2fs**, **chattr** et **lsattr**
`libext2fs`	Contient des routines pour permettre aux programmes niveau utilisateur de manipulater un système de fichiers `ext2`
`libss`	Utilisé par **debugfs**
`libuuid`	Contient des routines pour générer des identifieurs 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

grub	Le shell de commande pour Grub
grub-install	Installe GRUB sur le périphérique spécifié
grub-md5-crypt	Crypte un mot de passe au format MD5
grub-terminfo	Génère une commande terminfo à partir d'un nom terminfo. Il est utilisable si vous

	avez un terminal non usuel
mbchk	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

compress	Compresse et décompresse des fichiers
gunzip	Décompresse les fichiers gzip
gzexe	Crée des fichiers exécutables auto-extractibles
gzip	Compresse les fichiers données en utilisant le codage Lempel-Ziv (LZ77)

uncompress	Décompresse les fichiers compressés
zcat	Décompresse les fichiers gzip sur la sortie standard
zcmp	Lance **cmp** sur des fichiers compressés avec gzip
zdiff	Lance **diff** sur des fichiers compressés avec gzip
zegrep	Lance **egrep** sur des fichiers compressés avec gzip
zfgrep	Lance **fgrep** sur des fichiers compressés avec gzip
zforce	Force une extension `.gz` sur tous les fichiers donnés qui sont au format gzip, pour que **gzip** ne les compresse pas de nouveau ; ceci est utile quand les noms de fichiers sont tronqués lors d'un transfert de fichiers
zgrep	Lance **grep** sur des fichiers compressés avec gzip
zless	Lance **less** sur des fichiers compressés avec gzip
zmore	Lance **more** sur des fichiers compressés avec gzip
znew	Re-compresse des fichiers à partir du format **compress** vers le format **gzip**—`.Z` to `.gz`

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'identifieur 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

apropos	Cherche dans la base de données **whatis** et affiche les descriptions courtes des commandes systèmes contenant une chaîne donnée
makewhatis	Construit la base de données **whatis** ; elle lit toutes les pages man comprises dans `MANPATH` et, pour chaque page, écrit le nom et une courte description dans la base de données **whatis**
man	Formate et affiche la page man demandée en ligne
man2dvi	Convertit une page man au format dvi
man2html	Convertit une page man au format HTML
whatis	Recherche dans la base de données **whatis** 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
insmod	Installe un module chargeable dans le noyau en cours d'exécution
insmod.static	Une version compilée statiquement de **insmod**
lsmod	Liste les modules déjà chargés
modinfo	Examine un fichier objet associé à un module du noyau et affiche toute information qu'il peut récupérer
modprobe	Utilise un fichier de dépendances, créé par **depmod**, pour charger automatiquement les modules adéquats
rmmod	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

free	Indique le total de mémoire libre et utilisé sur le système à la fois pour la mémoire physique et pour la mémoire swap
kill	Envoie des signaux aux processus
pgrep	Recherche les processus suivant leur nom et autres attributs
pkill	Envoie des signaux aux processus suivant leur nom et autres attributs
pmap	Affiche le plan mémoire du processus désigné
ps	Donne un aperçu des processus en cours d'exécution
skill	Envoie des signaux aux processus correspondant à un critère donné
snice	Modifie les priorités des processus suivant le critère donné
sysctl	Modifie les paramètres du noyau en cours d'exécution
tload	Affiche un graphe de la charge système actuelle
top	Affiche une liste des processus demandant le maximum de ressources CPU ; il fournit un affichage agréable sur l'activité du processeur en temps réel
uptime	Affiche le temps d'exécution du système, le nombre d'utilisateurs connectés et les moyennes de charge système

vmstat Affiche les statistiques de mémoire virtuelle, donne des informations sur les processus, la mémoire, la pagination, le nombre de blocs en entrées/sorties, les échappements et l'activité CPU

w Affiche les utilisateurs actuellement connectés, où et depuis quand

watch Lance une commande de manière répétée, affichant le premier écran de sa sortie ; ceci vous permet de surveiller la sortie

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

6.53. 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, déplaçez-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

fuser	Indique les PID de processus utilisant les fichiers ou systèmes de fichiers donnés
killall	Tue les processus suivant leur nom. Il envoie un signal à tous les processus en cours
pstree	Affiche les processus en cours hiérarchiquement
pstree.x11	Identique à **pstree**, si ce n'est qu'il attend une confirmation avant de quitter

6.54. 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 paramétrages 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

chage Utilisé pour modifier le nombre maximum de jours avant d'obliger un changement de mot de passe

chfn Utilisé pour modifier le nom complet de l'utilisateur et quelques autres informations

chpasswd Utilisé pour mettre à jour les mots de passe d'une série de compte en une fois

chsh Utilisé pour modifier le shell de connexion par défaut d'un utilisateur

expiry vérifie et renforce la politique d'expiration des mots de passe

faillog Est utilisé pour examiner les traces d'échecs de connexions, pour configurer le nombre maximum d'échecs avant qu'un compte ne soit bloqué ou pour réinitialiser le nombre

	d'échecs
gpasswd	Est utilisé pour ajouter et supprimer des membres et des administrateurs aux groupes
groupadd	Crée un groupe avec le nom donné
groupdel	Supprime le groupe ayant le nom donné
groupmod	Est utilisé pour modifier le nom ou le GID du groupe
grpck	Vérifie l'intégrité des fichiers `/etc/group` et `/etc/gshadow`
grpconv	Ccrée ou met à jour le fichier shadow à partir du fichier group standard
grpunconv	Met à jour `/etc/group` à partir de `/etc/gshadow` puis supprime ce dernier
lastlog	Indique les connexions les plus récentes de tous les utilisateurs ou d'un utilisateur donné
login	Est utilisé par le système pour permettre aux utilisateurs de se connecter
logoutd	Est un démon utilisé pour renforcer les restrictions sur les temps et ports de connexion
mkpasswd	Crypte les mots de passe au hasard
newgrp	Est utilisé pour modifier le GID courant pendant une session de connexion
newusers	Est utilisé pour créer ou mettre à jour toute une série de comptes utilisateur en une fois
passwd	Est utilisé pour modifier le mot de passe d'un utilisateur ou d'un groupe
pwck	Vérifie l'intégrité des fichiers de mots de passe, `/etc/passwd` et `/etc/shadow`
pwconv	Crée ou met à jour le fichier de mots de passe shadow à partir du fichier password habituel
pwunconv	Met à jour `/etc/passwd` à partir de `/etc/shadow` puis supprime ce dernier
sg	Exécute une commande donnée lors de l'initialisation du GID de l'utilisateur à un groupe donné
su	Lance un shell en substituant les ID de l'utilisateur et du groupe
useradd	Crée un nouvel utilisateur avec le nom donné ou met à jour les informations par défaut du nouvel utilisateur
userdel	Supprime le compte utilisateur indiqué
usermod	Est utilisé pour modifier le nom de connexion de l'utilisateur, son UID (*User Identification*, soit Identification Utilisateur), shell, groupe initial, répertoire personnel et ainsi de suite
vigr	Édite les fichiers `/etc/group` et `/etc/gshadow`
vipw	Édite les fichiers `/etc/passwd` et `/etc/shadow`
`libshadow`	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 Sysklogd

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

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

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

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

6.55.3. Contenu de Sysklogd

Programmes installés: klogd et syslogd

Descriptions courtes

klogd	Un démon système pour intercepter et tracer les messages du noyau
syslogd	Trace les messages que les programmes systèmes donnent

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** envoit 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**. L 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

halt	Lance normalement **shutdown** avec l'option *-h*, sauf s'il est déjà au niveau d'exécution 0, puis il demande au noyau d'arrêter le système. Mais, tout d'abord, il note dans le fichier `/var/log/wtmp` que le système est en cours d'arrêt
init	Le premier processus à être exécuté lorsque le noyau a initialisé le matériel et qui prend la main sur le processus de démarrage et démarre tous les processus qui lui ont été indiqués
killall5	Envoie un signal à tous les processus sauf les processus de sa propre session, de façon à ne pas tuer le shell ayant lancé le script qui l'a appelé
last	Affiche le dernier utilisateur connecté (et déconnecté) en cherchant dans le fichier `/var/log/wtmp`. Il peut aussi afficher les démarrages et arrêts du système ainsi que les changements de niveaux d'exécution
lastb	Affiche les tentatives échouées de connexions tracées dans `/var/log/btmp`
mesg	Contrôle si les autres utilisateurs peuvent envoyer des messages au terminal de l'utilisateur courant
mountpoint	Vérifie si le répertoire est un point de montage
pidof	Indique le PID des programmes précisés
poweroff	Indique au noyau d'arrêter le système et de couper l'ordinateur (voir **halt**)
reboot	Indique au noyau de redémarrer le système (voir **halt**)
runlevel	Indique le niveau d'exécution actuel et précédent comme précisé dans l'enregistrement du dernier niveau d'exécution dans `/var/run/utmp`
shutdown	Arrête proprement le système en le signalant à tous les processus et à tous les utilisateur connectés
sulogin	Permet la connexion du superutilisateur. Il est normalement appelé par **init** lorsque le système passe en mono-utilisateur
telinit	indique à **init** dans quel niveau d'exécution entrer
utmpdump	Affiche le contenu du fichier de connexion donné dans un format plus agréable
wall	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 **udevd** 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 les 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

udevd Un démon qui réordonne les événements de montage à chaud avant de les soumettre à **udev**, évitant ainsi différentes conditions particulières

udevsend Délivre les événements de montage à chaud à **udevd**

udevstart Crée des nœuds périphériques dans `/dev` 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 **udev**

udevinfo Autorise les utilisateurs à envoyer des requêtes à la base de données **udev** 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 `sysfs` pour aider à la création des règles udev

udevtest Simule un lancement d'**udev** pour le périphérique donné et affiche le nom du nœud que le vrai **udev** aurait créé ou (pas dans LFS) le nom de l'interface réseau renommé

`/etc/udev` Contient les fichiers de configuration d'**udev**, 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, setsid, setterm, sfdisk, swapdev, swapoff (lien vers swapon), swapon, tunelp, ul, umount, vidmode (lien vers rdev), whereis et write

Descriptions courtes

agetty	Ouvre un port tty, demande un nom de connexion puis appelle le programme **login**
arch	Affiche l'architecture de la machine
blockdev	Permet aux utilisateurs d'appeler les ioctl d'un périphérique bloc à partir de la ligne de commande
cal	Affiche un calendrier simple
cfdisk	Manipule la table des partitions du périphérique donné
chkdupexe	Trouve les exécutables dupliqués
col	Filtre les retours chariot inversés
colcrt	filtrer la sortie de **nroff** pour les terminaux manquant de capacités comme le texte barré ou les demi-lignes
colrm	Filtre les colonnes données
column	Formate un fichier donné en de nombreuses colonnes
ctrlaltdel	Initialise la combinaison des touches Ctrl+Alt+Del pour une réinitialisation matérielle ou logicielle
cytune	Est utilisé pour paramétrer finement les pilotes de lignes séries des cartes Cyclades
ddate	Donne la date discordienne ou convertit la date grégorienne en une date discordienne
dmesg	Affiche les messages du noyau lors du démarrage
elvtune	Est utilisé pour configurer finement les performances et l'interactivité d'un périphérique bloc
fdformat	Réalise un formatage de bas niveau sur une disquette
fdisk	Est utilisé pour manipuler la table de partitions du périphérique donné
fsck.cramfs	Réalise un test de consistance sur le système de fichiers Cramfs du périphérique donné
fsck.minix	Réalise un test de consistence sur le système de fichiers Minix du périphérique donné
getopt	Analyse les options sur la ligne de commande donnée
hexdump	Affiche le fichier indiqué en hexadécimal ou dans un autre format donné
hwclock	Lit ou initialise l'horloge matériel, aussi appelée horloge RTC (*Real-Time Clock*, horloge à temps réel) ou horloge BIOS (*Basic Input-Output System*)
ipcrm	Supprime la ressource IPC donnée

ipcs	Fournit l'information de statut IPC
isosize	Affiche la taille d'un système de fichiers iso9660
line	Copie une simple ligne
logger	Enregistre le message donné dans les traces système
look	Affiche les lignes commençant avec la chaîne donnée
losetup	Initialiser et contrôle les périphériques loop
mcookie	génère des cookies magiques, nombres hexadécimaux aléatoires sur 128 bits, pour **xauth**
mkfs	Construit un système de fichiers sur un périphérique (habituellement une partition du disque dur)
mkfs.bfs	Crée un système de fichiers bfs de SCO (Santa Cruz Operations)
mkfs.cramfs	Crée un système de fichiers cramfs
mkfs.minix	Crée un système de fichiers Minix
mkswap	initialise le périphérique ou le fichier à utiliser comme swap
more	est un filtre pour visualiser un texte un écran à la fois
mount	Attache le système de fichiers du périphérique donné sur un répertoire spécifié dans le système de fichiers
namei	Affiche les liens symboliques dans les chemins donnés
pg	Affiche un fichier texte un écran à la fois
pivot_root	Fait en sorte que le système de fichiers donné soit le nouveau système de fichiers racine du processus actuel
ramsize	Initialise la taille du disque RAM dans une image amorçable
raw	Utilisé pour envoyer une requête et initialiser le périphérique racine et d'autres choses dans une image amorçable
rdev	Utilisé pour envoyer une requête et initialiser le périphérique racine et d'autres choses dans une image amorçable
readprofile	Lit les informations de profilage du noyau
rename	Renomme les fichiers donnés, remplaçant une chaîne donnée par une autre
renice	Modifie la priorité des processus exécutés
rev	Inverse les lignes d'un fichier donné
rootflags	Initialise les options racine d'une image amorçable
script	Crée un script à partir d'une session du terminal, de tout ce qui est affiché sur un terminal
setfdprm	Initialise les paramètres de disquette fournis par l'utilisateur
setsid	Lance le programme donné dans une nouvelle session
setterm	Initialise les attributs du terminal

sfdisk	Est un manipulateur de table de partitions disque
swapdev	Initialise le périphérique swap dans une image amorçable
swapoff	Désactive les périphériques et fichiers de pagination et de swap
swapon	Active les périphériques et fichiers de pagination et de swap, et liste les périphériques et fichiers en cours d'utilisation.
tunelp	Est utilisé pour paramétrer finement une imprimante ligne
ul	Un filtre pour traduire les souslignements en séquences d'échappement indiquant un souslignement pour le terminal utilisé
umount	Déconnecte un système de fichiers à partir de la hiérarchie de fichiers du système
vidmode	Initialise le mode vidéo d'une image amorçable
whereis	Affiche l'emplacement du binaire, les sources et la page de manuel de la commande donnée
write	Envoie un message à l'utilisateur donné *sauf si* 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 diminué d'environ 200 Mo en supprimant les symboles de débogage contenus dans les binaires et dans les bibliothèques. Ceci ne pose pas de problème autre que le fait de ne plus pouvoir les déboguer.

La plupart des personnes qui utilisent la commande mentionnée ci-dessous ne rencontrent aucune difficulté. Néanmoins, il est facile de faire une erreur de saisie et rendre le nouveau système complètement inutilisable, donc avant d'exécuter la commande **strip**, il est recommandé de faire une sauvegarde de l'état actuel.

Avant d'exécuter la suppression de ces symboles, faites particulièrement attention qu'aucun des binaires concernés ne sont en cours d'exécution. Si vous n'êtes pas sûr que l'utilisateur est entré dans chroot avec la commande donnée dans Section 6.3, « 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 traitées en toute sécurité :

```
/tools/bin/find /{,usr/}{bin,lib,sbin} -type f \
   -exec /tools/bin/strip --strip-debug '{}' ';'
```

Un grand nombre de fichiers seront rapportés comme ayant un format non reconnu. Ces messages d'avertissement indiquent que ces fichiers sont des scripts et non pas des binaires.

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

checkfs	Vérifie l'intégrité des systèmes de fichiers avant de les monter (avec l'exception des systèmes de fichiers journalisés ou réseau)
cleanfs	Supprime les fichiers qui ne devraient pas être conservés après un redémarrage, tels que ceux compris dans `/var/run/` et `/var/lock/` ; il re-crée `/var/run/utmp` et supprime les fichiers `/etc/nologin`, `/fastboot` et `/forcefsck` qui pourraient être présents
console	Charge la bonne table de correspondance du clavier ; il initialise aussi la police de l'écran
functions	Contient des fonctions communes, telles que la vérification d'erreurs et de statuts, utilisées par les différents scripts.
halt	Arrête le système
hotplug	Charge des modules pour les périphériques système
ifdown	Assiste le script network pour l'arrêt des périphériques réseaux
ifup	Assiste le script network pour le démarrage des périphériques réseaux
localnet	Configure le nom d'hôte du système et le périphérique de boucle locale
mountfs	Monte tous les systèmes de fichiers, sauf ceux marqués *noauto* et les systèmes réseaux
mountkernfs	Monte les systèmes de fichiers virtuels fournies par le noyau, tels que `proc`
network	Configure les interfaces réseaux, telles que les cartes réseaux, et configure la passerelle par défaut (lorsque c'est applicable)
rc	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

reboot	Redémarre le système
sendsignals	S'assure que chaque processus est terminé avant que le système redémarre ou ne s'arrête
setclock	Réinitialise l'horloge noyau avec l'heure locale au cas où l'horloge matérielle n'est pas en temps UTC
static	Fournit les fonctionnalités nécessaires à l'affectation d'une adresse statique IP (Internet Protocol) vers une interface réseau
swap	Active et désactive les fichiers swap et les partitions
sysklogd	Lance et arrête les démons des journaux système et noyau
template	Un modèle pour créer des scripts de démarrage personnalisés pour d'autres démons
udev	Prépare le répertoire `/dev` et lance Udev.

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 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 adopté par `devfs` était la façon dont il gérait la détection, la création et le nommage des périphériques. Ce dernier problème, le nommage des périphériques, était peut-être le plus critique. Il est généralement accepté que s'il est possible de configurer les noms des périphériques, alors la politique de nommage des périphériques revient à l'administrateur du système, et du coup n'est 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 une 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 paquetage LFS-Bootscripts. Si **udev** ne peut as 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'oelig;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 paquetage 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* (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* (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 rappeler 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 être 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 paramétrages 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 paramétrages 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 '\W14\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 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 paramétrages globaux.

Un shell interactif de connexion est lancé après une connexion réussie, en utilisant **/bin/login**, par la lecture du fichier `/etc/passwd`. Un shell interactif sans connexion est lancé en ligne de commande (c'est-à-dire `[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 abec 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 suivan te 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 fichier 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 u tilisés par les fournisseurs d'accès à Internet DSL et cable. 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 identifieurs 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 lontemps. À cause de cela, il existe une chance pour que l'identifieur 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

`config-2.6.11.12`	Contient toutes les sélections de la configuration pour le noyau
`noyau`	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.
`System.map-2.6.11.12`	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 `hdb` 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

ABI Application Binary Interface

ALFS Automated Linux From Scratch

ALSA Advanced Linux Sound Architecture

API Application Programming Interface

ASCII American Standard Code for Information Interchange

BIOS Basic Input/Output System

BLFS Beyond Linux From Scratch

BSD Berkeley Software Distribution

chroot change root

CMOS Complementary Metal Oxide Semiconductor

COS Class Of Service

CPU Central Processing Unit

CRC Cyclic Redundancy Check

CVS Concurrent Versions System

DHCP Dynamic Host Configuration Protocol

DNS Domain Name Service

EGA Enhanced Graphics Adapter

ELF Executable and Linkable Format

EOF End of File

EQN equation

EVMS Enterprise Volume Management System

ext2 second extended file system

FAQ Frequently Asked Questions

FHS Filesystem Hierarchy Standard

FIFO First-In, First Out

FQDN Fully Qualified Domain Name

FTP File Transfer Protocol

GB Gibabytes

GCC GNU Compiler Collection

GID Group Identifier

GMT Greenwich Mean Time

GPG GNU Privacy Guard

HTML Hypertext Markup Language

IDE Integrated Drive Electronics

IEEE Institute of Electrical and Electronic Engineers

IO Input/Output

IP Internet Protocol

IPC Inter-Process Communication

IRC Internet Relay Chat

ISO International Organization for Standardization

ISP Internet Service Provider

KB Kilobytes

LED Light Emitting Diode

LFS Linux From Scratch

LSB Linux Standards Base

MB Megabytes

MBR Master Boot Record

MD5 Message Digest 5

NIC Network Interface Card

NLS Native Language Support

NNTP Network News Transport Protocol

NPTL Native POSIX Threading Library

OSS Open Sound System

PCH Pre-Compiled Headers

PCRE Perl Compatible Regular Expression

PID Process Identifier

PLFS Pure Linux From Scratch

PTY pseudo terminal

QA Quality Assurance

QOS Quality Of Service

RAM Random Access Memory

RPC Remote Procedure Call

RTC Real Time Clock

SBU Standard Build Unit

SCO The Santa Cruz Operation

SGR Select Graphic Rendition

SHA1 Secure-Hash Algorithm 1

SMP Symmetric Multi-Processor

TLDP The Linux Documentation Project

TFTP Trivial File Transfer Protocol

TLS Thread-Local Storage

UID User Identifier

umask user file-creation mask

USB Universal Serial Bus

UTC Coordinated Universal Time

UUID Universally Unique Identifier

VC Virtual Console

VGA Video Graphics Array

VT Virtual Terminal

Annexe B. 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

Index

Paquets

Programmes

Bibliotheques

Scripts

Autres