Linux From Scratch

Version 12.4

Publié le 01 septembre 2025

Créé par Gerard Beekmans
Rédacteur en chef : Bruce Dubbs

Linux From Scratch: Version 12.4: Publié le 01 septembre 2025

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

Table des matières

Préface

Avant-propos

Mon parcours pour apprendre et mieux comprendre Linux a débuté en 1998. Je venais d'installer ma première distribution Linux et je fus rapidement intrigué par l'ensemble du concept et la philosophie sous-jacente de Linux.

Il y a toujours bien des manières d'accomplir une seule tâche. Il en est de même pour les distributions Linux. Un grand nombre existent depuis des années. Certaines existent encore, certaines se sont transformées en quelque chose d'autre, tandis que d'autres encore ont été reléguées à nos souvenirs. Elles font toutes les choses différemment pour s'adapter au besoin de leur public. Vu qu'il existait énormément de manières différentes d'atteindre le même objectif, je me rendis compte que je n'étais plus obligé de me limiter à une organisation en particulier. Avant de découvrir Linux, on supportait simplement les problèmes dans d'autres systèmes d'exploitation puisqu'on n'avait pas le choix. Cela valait ce que ça valait, que cela nous plaise ou non. Avec Linux, le concept du choix a commencé à émerger. Si vous n'aimiez pas quelque chose, vous étiez libres, voire encouragés à le modifier.

J'ai essayé un certain nombre de distributions et n'ai pas pu me décider pour l'une d'entre elles. C'étaient de bons systèmes, chacun à sa façon. Ce n'était plus une question de bonne ou mauvaise qualité. C'était devenu une question de goût personnel. Avec tout ce choix disponible, il devenait clair qu'il n'y aurait pas un seul système qui serait parfait pour moi. Donc je résolus de créer mon propre système Linux qui correspondrait totalement à mes préférences personnelles.

Pour que ce soit vraiment mon propre système je résolus de compiler tout à partir du code source au lieu d'utiliser des paquets de binaires pré-compilés. Ce système Linux « parfait » aurait les forces de plusieurs systèmes sans leurs faiblesses ressenties. De prime abord, l'idée était décourageante. Je restais sceptique à la pensée de pouvoir construire un tel système.

Après avoir rencontré quelques problèmes comme des dépendances circulaires et des erreurs à la compilation, j'ai finalement construit un système Linux entièrement personnalisé. Il était totalement opérationnel et parfaitement utilisable comme n'importe quel autre système Linux du moment. Mais c'était ma propre création. C'était très satisfaisant d'avoir concocté un tel système moi-même. Créer chaque morceau de logiciel par moi-même aurait été la seule option encore plus satisfaisante. C'était là la meilleure alternative.

Alors que je partageais mes objectifs et mes expériences avec d'autres membres de la communauté Linux, il devint manifeste qu'il y avait un intérêt soutenu concernant ces idées. Il devint rapidement clair que de tels systèmes Linux personnalisés satisfaisaient non seulement les exigences des utilisateurs mais servaient aussi d'une opportunité idéale d'apprentissage pour les programmeurs et les administrateurs systèmes, afin d'améliorer leurs compétences (existantes) sous Linux. De cet intérêt est né le projet *Linux From Scratch*..

Ce livre *Linux From Scratch* est le cœur de ce projet. Il fournit la base et les instructions qui vous sont nécessaires pour concevoir et construire votre propre système. Bien que ce livre fournisse un modèle qui aboutira à un système qui fonctionne correctement, vous êtes libres de modifier les instructions pour les adapter à vos envies, ce qui fait partie des finalités importantes du projet après tout. Vous gardez le contrôle ; nous vous donnons simplement un coup de main pour débuter votre propre parcours.

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

--
Gerard Beekmans
gerard@linuxfromscratch.org

Public visé

Beaucoup de raisons peuvent vous pousser à vouloir lire ce livre. Une des questions que beaucoup de monde se pose est « pourquoi se fatiguer à construire à la main un système Linux de A à Z alors qu'il suffit de télécharger et installer une distribution existante ? »

Vous aider à apprendre comment fonctionne un système Linux de l'intérieur est l'une des raisons importantes de l'existence de ce projet. Construire un système LFS permet de démontrer ce qui fait que Linux fonctionne, et comment les choses interagissent et dépendent les unes des autres. L'une des meilleures choses que l'expérience de cet apprentissage peut vous apporter est la capacité à personnaliser un système Linux afin qu'il soit à votre goût et réponde à vos besoins.

Un autre avantage clé de LFS est qu'il vous permet d'avoir plus de contrôle sur votre système sans avoir à dépendre d'une implémentation créée par quelqu'un d'autre. Avec LFS, *vous* êtes maintenant au volant et vous êtes capable de décider de chaque aspect du système.

LFS vous permet de créer des systèmes Linux très compacts. Lors de l'installation d'autres distributions, vous êtes souvent obligé d'installer de nombreux programmes que vous n'utiliserez ni ne comprendrez probablement jamais. Ces programmes gaspillent des ressources. Vous pourriez répondre qu'avec les disques durs et les processeurs d'aujourd'hui, les ressources ne sont plus un problème. Pourtant, vous êtes parfois contraint par des questions d'espace, ou d'autres limitations. Pensez aux CD, clés USB amorçables et aux systèmes embarqués. Ce sont des domaines où LFS peut être avantageux.

Un autre avantage d'un système Linux personnalisé est un surcroît de sécurité. En compilant le système complet à partir du code source, vous avez la possibilité de tout vérifier et d'appliquer tous les correctifs de sécurité désirés. Il n'est plus nécessaire d'attendre que quelqu'un d'autre vous fournisse les paquets binaires pour réparer une faille de sécurité. À moins que vous n'examiniez vous-même le correctif et que vous ne l'appliquiez vous-même, vous n'avez aucune garantie que le nouveau paquet binaire ait été compilé correctement et qu'il corrige bien le problème.

Le but de *Linux From Scratch* est de construire les fondations d'un système complet et utilisable. Si vous ne souhaitez pas construire votre propre système à partir de rien, vous pourriez cependant bénéficier des informations contenues dans ce livre.

Il existe trop de bonnes raisons de construire votre système LFS pour pouvoir toutes les lister ici. En fin de compte, l'apprentissage est de loin la raison la plus puissante. En continuant dans votre expérience de LFS, vous découvrirez le pouvoir réel que donnent l'information et la connaissance.

Architectures cibles de LFS

Les principales architectures cibles de LFS sont les processeurs AMD ou Intel x86 (32 bits) et x86_64 (64 bits). En même temps, les instructions de ce livre fonctionnent également, avec quelques modifications, sur les processeurs Power PC et ARM. Pour construire un système qui utilise un de ces processeurs, le prérequis principal, supplémentaire à ceux des pages suivantes, est la présence d'un système Linux grâce à une installation précédente de LFS, Ubuntu, Red Hat/Fedora, SuSE, ou une autre distribution ciblant l'architecture de vote processeur. Remarquez aussi que vous pouvez installer et utiliser un système 32 bits en tant que système hôte sur un système AMD ou Intel 64 bits.

Le gain obtenu en compilant sur un système 64 bits comparé à un système 32 bits est minimal. Par exemple, dans le test de la construction de LFS-9.1 sur un système basé sur un processeur Core i7-4790, nous avons relevé les statistiques suivantes :

```
Temps de construction de l'architecture     Taille de la construction
32 bit       239,9 minutes  3,6 Go
64 bit       233,2 minutes  4,4 Go
```

Comme vous pouvez le constater, sur le même matériel, la construction 64 bits est seulement 3 % plus rapide et 22 % plus grosse que la construction en 32 bits. Si vous voulez utiliser LFS pour un serveur LAMP, ou un pare-feu, un processeur 32 bits est largement suffisant. Au contraire, plusieurs paquets dans BLFS ont maintenant besoin de plus de 4 Go de RAM pour être construits ou lancés. Si vous voulez utiliser LFS sur un ordinateur de bureau, les auteurs de LFS recommandent de construire un système 64 bits.

La construction 64 bits par défaut qui résulte de LFS est considérée comme un système « pur » 64 bits. C'est-à-dire qu'elle ne prend en charge que les exécutables en 64 bits. La construction d'un système « multi-lib » implique la construction de beaucoup d'applications à deux reprises, une fois pour le système 32 bits et une fois pour le système 64 bits. Ceci n'est pas pris en charge par LFS car cela interférerait avec l'objectif pédagogique visant à fournir les instructions nécessaires à un simple système Linux de base. Certains éditeurs de LFS et BLFS maintiennent un fork multilib de LFS, accessible sur *https://www.linuxfromscratch.org/~thomas/multilib/index.html*. Toutefois cela reste un sujet avancé.

Prérequis

Construire un système LFS n'est pas une tâche facile. Cela requiert un certain niveau de connaissance en administration de système Unix pour résoudre les problèmes et exécuter correctement les commandes listées. En particulier, vous devriez au minimum déjà savoir comment utiliser la ligne de commande (le shell) pour copier et déplacer des fichiers et des répertoires, pour lister le contenu de répertoires et de fichiers, et pour changer de répertoire. Il est aussi attendu que vous disposiez d'une connaissance raisonnable de l'utilisation et de l'installation de logiciels Linux.

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

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

- Software-Building-HOWTO *https://tldp.org/HOWTO/Software-Building-HOWTO.html*

  C'est un guide complet sur la construction et l'installation « générique » de logiciels Unix sous Linux. Bien qu'il ait été écrit il y a longtemps, il offre encore un bon résumé des techniques de base requises pour construire et installer un logiciel.

- Beginner's Guide to Installing from Source (Guide de l’installation à partir des sources pour le débutant) *https://moi.vonos.net/linux/beginners-installing-from-source/*

  Ce guide propose un bon résumé des compétences et des connaissances techniques de base nécessaires à la construction de logiciels à partir du code source.

LFS et les standards

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

- *POSIX.1-2008.*
- *Filesystem Hierarchy Standard (FHS) version 3.0*
- *Linux Standard Base (LSB) version 5.0 (2015)*

  La LSB comporte quatre spécifications distinctes : le cœur, le bureau, les langages et l'impression. Certaines parties des spécifications du cœur et du bureau comportent des exigences qui s'appliquent à la construction. Il y a aussi deux spécifications pour l'évaluation : Gtk3 et les graphismes. LFS s'applique à respecter les constructions IA32 (32-bit x86) ou AMD64 (x86_64) qui ont été évoquées dans la section précédente.

Note

Les exigences de la LSB ne font pas l'unanimité. Leur objectif principal est de s'assurer que les logiciels propriétaires pourront être installés et lancés correctement sur un système conforme. Comme LFS est basée sur le code source, les utilisateurs ont un contrôle total des paquets qui les intéressent et certains choisissent de ne pas installer certains paquets qui sont cités dans la LSB.

Il est possible de créer de toute pièce un système complet qui soit capable de réussir les tests de certification LSB « de zéro », mais cela requiert des paquets supplémentaires qui vont au-delà des objectifs de LFS. Vous trouverez les instructions d'installation de certains de ces paquets supplémentaires dans BLFS.

Paquets fournis par LFS requis pour satisfaire les exigences de la LSB

Cœur de la LSB :	Bash, Bc, Binutils, Coreutils, Diffutils, File, Findutils, Gawk, GCC, Gettext, Glibc, Grep, Gzip, M4, Man-DB, Procps, Psmisc, Sed, Shadow, SysVinit, Tar, Util-linux, Zlib
LSB pour les bureaux :	Aucun
LSB pour les langages :	Perl
LSB pour l'impression :	Aucun
LSB pour Gtk3 et LSB pour les graphismes (évaluation):	Aucun

Paquets fournis par BLFS requis pour satisfaire les exigences de la LSB

Cœur de la LSB :	At, Batch (part d'At), BLFS Bash Startup Files, Cpio, Ed, Fcrontab, LSB-Tools, NSPR, NSS, Linux-PAM, Pax, Sendmail (ou Postfix ou Exim), Time
LSB pour les bureaux :	Alsa, ATK, Cairo, Desktop-file-utils, Freetype, Fontconfig, Gdk-pixbuf, Glib2, GLU, Icon-naming-utils, Libjpeg-turbo, Libxml2, Mesa, Pango, Xdg-utils, Xorg
LSB pour les langages :	Libxml2, Libxslt
LSB pour l'impression :	CUPS, Cups-filters, Ghostscript, SANE
LSB pour Gtk3 et LSB pour les graphismes (évaluation):	GTK+3

Les composants qui ne sont ni fournis ni fournis de manière facultatives, ni par LFS ni par BLFS, mais qui sont requis pour satisfaire les exigences de la LSB

Cœur de la LSB :	**install_initd**, `libcrypt.so.1` (peut être fourni avec des instructions facultatives du paquet Libxcrypt dans LFS), `libncurses.so.5` (peut être fourni avec des instructions facultatives du paquet Ncurses de LFS), `libncursesw.so.5` (mais `libncursesw.so.6` est fourni par le paquet Ncurses de LFS)
LSB pour les bureaux :	`libgdk-x11-2.0.so` (mais `libgdk-3.so` est fourni par le paquet GTK+-3 de BLFS), `libgtk-x11-2.0.so` (mais `libgtk-3.so` et `libgtk-4.so` sont fournis par les paquets GTK+-3 et GTK+-4 de BLFS), `libpng12.so` (mais `libpng16.so` est fourni par le paquet Libpng de BLFS),

	`libQt*.so.4` (mais `libQt6*.so.6` sont fournis par le paquet Qt6 de BLFS), `libtiff.so.4` (mais `libtiff.so.6` est fourni par le paquet Libtiff de BLFS)
LSB pour les langages :	**/usr/bin/python** (la LSB nécessite Python2 mais LFS et BLFS ne fournissent que Python3)
LSB pour l'impression :	Aucun
LSB pour Gtk3 et LSB pour les graphismes (évaluation):	`libpng15.so` (mais `libpng16.so` est fourni par le paquet Libpng de BLFS)

Raison de la présence des paquets dans le livre

Comme indiqué plus haut, le but de LFS est de construire les fondations complètes et utilisables d'un système. Il inclut tous les paquets nécessaires pour être répliqué tout en fournissant une base relativement minimale vous permettant de personnaliser un système plus complet basé sur les choix de l'utilisateur. Cela ne veut pas dire que LFS est le plus petit système possible. Plusieurs paquets importants sont inclus et ne sont pas absolument indispensables. Les listes ci-dessous documentent la raison pour laquelle chaque paquet se trouve dans le livre.

- Acl

  Ce paquet contient des outils d'administration des listes de contrôle d'accès, utilisées pour définir plus finement les droits d'accès de votre choix pour les fichiers et les répertoires.

- Attr

  Ce paquet contient des programmes d'administration des attributs étendus sur les objets d'un système de fichiers.

- Autoconf

  Le paquet Autoconf contient des programmes produisant des scripts shell qui configurent automatiquement le code source à partir du modèle fourni par le développeur. Il est souvent requis pour reconstruire un paquet après une mise à jour des procédures de construction.

- Automake

  Ce paquet contient des programmes pour générer des Makefile à partir d'un modèle. Il est souvent requis pour reconstruire un paquet après une mise à jour des procédures de construction.

- Bash

  Ce paquet satisfait une exigence du cœur de la LSB pour fournir une interface Bourne Shell au système. Il a été choisi parmi d'autres shells du fait de son utilisation répandue et de ses fonctionnalités étendues au-delà des fonctions d'un shell de base.

- Bc

  Ce paquet fournit un langage de traitement numérique à précision arbitraire. Il satisfait une exigence utilisée pour la construction du noyau Linux.

- Binutils

  Ce paquet contient un éditeur de liens, un assembleur et d'autres outils de gestion des fichiers objets. Les programmes de ce paquet sont nécessaires pour compiler la plupart des paquets d'un système LFS et au-delà.

- Bison

  Ce paquet contient la version GNU de yacc (*Yet Another Compiler Compiler*, encore un nouveau compilateur de compilateur) requis pour construire plusieurs autres programmes de LFS.

- Bzip2

Ce paquet contient des programmes de compression et de décompression de fichiers. Il est nécessaire pour décompresser plusieurs paquets de LFS.

- Check

  Ce paquet contient un harnais de tests pour d'autres programmes.

- Coreutils

  Ce paquet contient un certain nombre de paquets essentiels pour visualiser et manipuler des fichiers et des répertoires. Ces programmes sont nécessaires pour la gestion de fichiers en ligne de commande et ils sont nécessaires pour les procédures d'installation de chaque paquet de LFS.

- DejaGNU

  Ce paquet fournit un cadriciel pour tester d'autres programmes.

- Diffutils

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

- E2fsprogs

  Ce paquet contient les outils de gestion des systèmes de fichiers ext2, ext3 et ext4. Ce sont les systèmes de fichiers les plus courants et les plus largement testés, parmi ceux pris en charges par Linux.

- Expat

  Ce paquet contient une bibliothèque d'analyse XML relativement petite. Il est exigé par le module Perl XML::Parser.

- Expect

  Ce paquet contient un programme pour réaliser des dialogues scriptés avec d'autres programmes interactifs. Il est souvent utilisé pour tester d'autres paquets.

- File

  Ce paquet contient un outil pour déterminer le type d'un ou plusieurs fichiers donnés. Quelques paquets en ont besoin dans leur script de construction.

- Findutils

  Ce paquet contient des programmes pour rechercher des fichiers dans un système de fichiers. Il est utilisé dans les scripts de construction de nombreux paquets.

- Flex

  Ce paquet contient un outil de génération de programmes qui reconnaît des modèles de texte. C'est la version GNU du programme lex (*lexical analyzer*, analyseur lexical). Il est nécessaire pour construire plusieurs paquets LFS.

- Gawk

  Ce paquet contient des programmes de manipulation de fichiers texte. C'est la version GNU du programme awk (Aho-Weinberg-Kernighan). Il est utilisé dans les scripts de construction de nombreux autres paquets.

- GCC

  Ce paquet est le *Gnu Compiler Collection*. Il contient les compilateurs C et C++ ainsi que d'autres qui ne sont pas construits dans LFS.

- GDBM

Ce paquet contient la bibliothèque *GNU Database Manager* (gestionnaire de base de données GNU). Il est utilisé par un autre paquet de LFS : Man-DB.

- Gettext

  Ce paquet contient des outils et des bibliothèques pour l'internationalisation et la traduction de nombreux paquets.

- Glibc

  Ce paquet contient la bibliothèque C principale. Les programmes Linux ne peuvent pas s'exécuter sans elle.

- GMP

  Ce paquet contient des bibliothèques mathématiques qui fournissent des fonctions utiles pour de l'arithmétique en précision arbitraire. Il est nécessaire pour construire GCC.

- Gperf

  Ce paquet contient un programme qui génère une fonction de hachage parfaite à partir d'un trousseau. Il est exigé par Udev.

- Grep

  Ce paquet contient des programmes de recherche au sein de fichiers. Ces programmes sont utilisés par la plupart des scripts de construction des paquets.

- Groff

  Le paquet Groff contient des programmes de formatage de texte. Une des fonctions importantes de ces programmes est le formatage des pages de manuels.

- GRUB

  Ce paquet est le chargeur *Grand Unified Boot*. Ce n'est pas le seul chargeur d'amorçage disponible, mais c'est le plus flexible.

- Gzip

  Ces paquets contiennent des programmes de compression et de décompression de fichiers. Il est nécessaire pour décompresser de nombreux paquets sur LFS.

- Iana-etc

  Ce paquet fournit des données pour des services et des protocoles réseau. Il est nécessaire pour activer les bonnes fonctionnalités de réseau.

- Inetutils

  Ce paquet contient des programmes d'administration réseau de base.

- Intltool

  Ce paquet contient des outils pour extraire des chaînes traduisibles de fichiers sources.

- IProute2

  Ce paquet contient des programmes pour du réseau de base ou avancé en IPv4 et IPv6. Il a été choisi parmi les paquets d'outils réseau courants (net-tools) pour ses fonctionnalités IPv6.

- Kbd

  Ce paquet contient des fichiers de tables de touches, des outils claviers pour les claviers non américains et un certain nombre de polices pour console.

- Kmod

Ce paquet contient des programmes nécessaires pour administrer les modules du noyau Linux.

- Less

  Ce paquet contient un très bon visualiseur de texte qui permet le défilement vers le haut ou vers le bas lors de la visualisation d'un fichier. Il est aussi utilisé par Man-DB pour visualiser des pages de manuels.

- Libcap

  Ce paquet implémente les interfaces au niveau utilisateur avec les possibilités POSIX 1003.1e disponibles dans les noyaux Linux.

- Libelf

  Le projet elfutils fournit des bibliothèques et des outils pour les fichiers ELF et le format de données DWARF. La plupart des utilitaires de ce paquet sont disponibles dans d'autres paquets mais la bibliothèque est requise pour construire le noyau Linux avec la configuration par défaut (et la plus efficace).

- Libffi

  Ce paquet implémente une interface de programmation portable et haut-niveau pour diverses conventions d'appel. Certains programmes peuvent ne pas savoir à la compilation les arguments à passer à une fonction. Par exemple, il se peut qu'un interpréteur n'apprenne le nombre et le type des arguments utilisés pour appeler une fonction donnée qu'à l'exécution. Libffi peut être utilisée dans ces programmes pour fournir une passerelle entre l'interpréteur et le code compilé.

- Libpipeline

  Le paquet Libpipeline contient une bibliothèque pour manipuler des files (pipelines) de sous-processus de façon flexible et commode. Il est requis par le paquet Man-DB.

- Libtool

  Ce paquet contient le script générique de prise en charge des bibliothèques de GNU. Il englobe la complexité de l'utilisation des bibliothèques partagées dans une interface cohérente et portable. Il est exigé par les suites de tests d'autres paquets de LFS.

- Libxcrypt

  Ce paquet fournit la bibliothèque `libcrypt` requise par plusieurs paquets (notamment, Shadow) pour hasher les mots de passe. Il remplace l'implémentation obsolète `libcrypt` de Glibc.

- Noyau Linux

  Ce paquet est le système d'exploitation. C'est Linux dans l'environnement GNU/Linux.

- M4

  Ce paquet contient un traitement de macros textuelles générales utile en tant qu'outil de construction d'autres programmes.

- Make

  Ce paquet contient un programme de gestion de la construction des paquets. Il est requis par presque tous les paquets de LFS.

- Man-DB

  Ce paquet contient des programmes de recherche et de visualisation de pages de manuels. Il a été préféré au paquet man du fait de fonctionnalités d'internationalisation supérieures. Il fournit le programme man.

- Man-pages

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

- Meson

  Ce paquet fournit un outil logiciel pour automatiser la construction de logiciels. Le but principal de Meson est de minimiser le temps que les développeurs passent à configurer leur système de construction. Il est requis pour construire Systemd, ainsi que de nombreux paquets de BLFS.

- MPC

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

- MPFR

  Ce paquet contient des fonctions pour des maths à précision multiple. Il est exigé par GCC.

- Ninja

  Ce paquet contient un petit système de construction qui met l'accent sur la vitesse. Ses fichiers d'entrées sont prévus pour être générés par un système de construction de plus haut niveau, et il est prévu pour lancer des constructions aussi rapidement que possible. Ce paquet est requis par Meson.

- Ncurses

  Le paquet Ncurses contient les bibliothèques de gestion des écrans type caractère, indépendant des terminaux. Il est souvent utilisé pour fournir le contrôle du curseur dans un système en menus. Il est exigé par un certain nombre de paquets de LFS.

- Openssl

  Ce paquet fournit les outils de gestion et les bibliothèques liées à la cryptographie. Ils sont utiles pour fournir des fonctions de cryptographies à d'autres paquet, dont le noyau Linux.

- Patch

  Ce paquet contient un programme pour modifier ou créer des fichiers en appliquant un fichier de *correctif* créé en général par le programme diff. Il est requis par la procédure de construction de plusieurs paquets LFS.

- Perl

  Ce paquet est un interpréteur du langage PERL. Il est nécessaire pour l'installation et les suites de tests de plusieurs paquets LFS.

- Pkgconf

  Ce paquet contient un programme qui aide à configurer les drapeaux du compilateur et de l'éditeur de liens pour les bibliothèques de développement. Le programme peut être utilisé comme remplaçant direct de **pkg-config**, qui est requis par le système de construction de nombreux paquets. Il est plus activement maintenu et un peu plus rapide que le paquet Pkg-config original.

- Procps-NG

  Ce paquet contient des programmes de surveillance des processus. Ces programmes sont utiles pour l'administration système et ils sont aussi utilisés par les scripts de démarrage LFS.

- Psmisc

  Ce paquet contient des programmes d'affichage d'informations sur les processus en cours d'exécution. Ces programmes sont utiles pour l'administration système.

- Python 3

  Ce paquet fournit un langage interprété dont la philosophie de conception valorise la lisibilité du code.

- Readline

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

- Sed

  Ce paquet permet de saisir du texte sans ouvrir le fichier dans un éditeur de texte. Il est aussi requis par la plupart des scripts de configuration des paquets LFS.

- Shadow

  Ce paquet contient des programmes de gestion sécurisée des mots de passe.

- Sysklogd

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

- SysVinit

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

- Udev

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

- Tar

  Ce paquet fournit des fonctionnalités d'archivage et d'extraction de pratiquement tous les paquets utilisés dans LFS.

- Tcl

  Ce paquet contient le *Tool Command Language* utilisé dans beaucoup de suites de tests des paquets LFS.

- Texinfo

  Ce paquet contient des programmes de lecture, d'écriture et de conversion de pages info. Il est utilisé dans les procédures d'installation de beaucoup de paquets LFS.

- Util-linux

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

- Vim

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

- Wheel

  Ce paquet contient un module Python qui est l'implémentation de référence du standard de gestion des paquets Python wheel.

- XML::Parser

  Ce paquet est un module Perl qui interagit avec Expat.

- XZ Utils

  Ce paquet contient des programmes de compression et de décompression de fichiers. Il offre la compression la plus haute disponible et il est utile pour la décompression des paquets au format XZ ou LZMA.

- Zlib

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

- Zstd

Ce paquet contient des routines de compression et de décompression utilisées par quelques programmes. Il fournit de forts taux de compression et une large gamme de compromis entre compression et vitesse.

Typographie

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

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

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

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

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

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

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

Ce style de texte (texte à largeur fixe) montre une sortie d'écran, généralement le résultat de commandes. Si vous lisez le livre au format HTML (plutôt que le format PDF), le texte devrait être bleu.

Le texte à largeur fixe est aussi utilisé pour afficher des noms de fichiers, comme `/etc/ld.so.conf`.

Note

Configurez votre navigateur pour afficher les textes à chasse fixe avec une bonne police à chasse fixe, avec laquelle vous pouvez clairement distinguer les caractères dans `Il1` ou `O0`.

Mise en évidence

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

https://www.linuxfromscratch.org/

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

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

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

<TEXTE À REMPLACER>

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

[TEXTE FACULTATIF]

Ce format est utilisé pour intégrer du texte qui est facultatif.

passwd(5)

Ce format est utilisé pour faire référence à une page de manuel (man) spécifique. Le nombre entre parenthèses indique une section spécifique à l'intérieur des manuels. Par exemple, **passwd** a deux pages de manuel. Pour les instructions d'installation de LFS, ces deux pages de manuels seront situées dans `/usr/share/man/man1/passwd.1` et `/usr/share/man/man5/passwd.5`. Quand le livre utilise *passwd(5)*, il fait spécifiquement référence à `/usr/share/man/man5/passwd.5`. **man passwd** affichera la première page de manuel qu'il trouvera et qui correspond à « passwd », `/usr/share/man/man1/passwd.1`. Dans cet exemple, vous devrez exécuter **man 5 passwd** pour lire cette page spécifique. Il devrait être noté que la plupart des pages de man n'ont pas de nom de page dupliqué dans les différentes sections. Du coup, **man** ***<nom du programme>*** est généralement suffisant. Dans le livre LFS, ces références aux pages de manuel sont des liens hypertextes. Cliquer sur une telle référence ouvrira donc la page de manuel rendue au format HTML sur *les pages de manuel de Arch Linux*.

Structure

Ce livre est divisé en plusieurs parties.

Partie I — Introduction

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

Partie II — Préparation de la construction

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

Partie III — Construction des outils croisés LFS et des outils temporaires

La troisième partie fournit des instructions pour construire les outils requis pour la construction du système LFS final.

Partie IV — Construction du système LFS

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

Partie V — Annexes

La cinquième partie fournit des informations sur le livre lui-même, en particulier les acronymes et les termes utilisés, les remerciements, les dépendances des paquets, une liste des scripts de démarrage de LFS, la licence pour redistribuer ce livre et un catalogue complet des paquets, des programmes, des bibliothèques et des scripts.

Errata et annonces de sécurité

Les logiciels utilisés pour créer un système LFS sont constamment mis à jour et améliorés. Des messages d'avertissement pour la sécurité et des corrections de bogues pourraient survenir après la sortie du livre LFS. Afin de vérifier si les versions du paquet ou les instructions de cette version de LFS ont besoin de modifications pour corriger des vulnérabilités en termes de sécurité ou toute autre correction de bogue, merci de visiter *https://www.linuxfromscratch.org/lfs/errata/12.4/* avant de commencer votre construction. Vous devez noter toutes les modifications et les appliquer à la section correspondante du livre pendant votre progression lors de la construction du système LFS.

En plus, les rédacteurs de Linux From Scratch maintiennent une liste des vulnérabilités de sécurité découvertes *après* la publication du livre. Avant de continuer la construction de LFS, vérifiez la liste des vulnérabilités actives disponible sur *https://www.linuxfromscratch.org/lfs/advisories/*. Lors de la construction du système LFS, appliquez les modifications suggérées dans les annonces de chacune des sections importantes du livre. Si vous utilisez LFS comme un vrai système de bureau ou de serveur, continuez à prendre en compte les annonces et à corriger les vulnérabilités même après la construction du système LFS.

Partie I. Introduction

Chapitre 1. Introduction

1.1. Comment construire un système LFS

Le système LFS sera construit avec une distribution Linux déjà installée (telle que Debian, OpenMandriva, Fedora ou openSUSE). Ce système Linux existant (l'hôte) sera utilisé comme point de départ pour fournir les 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 pour disposer de ces outils.

Note

On peut installer une distribution Linux de nombreuses manières et les paramètres par défaut ne sont en général pas optimisés pour construire un système LFS. Consultez *https://www.linuxfromscratch.org/hints/downloads/files/partitioning-for-lfs.txt* pour trouver des suggestions de configuration pour des distributions commerciales.

Au lieu de l'installation d'une distribution séparée complète sur votre machine, vous pouvez utiliser le LiveCD d'une distribution commerciale.

Le Chapitre 2 de ce livre décrit comment créer une nouvelle partition Linux native et un système de fichiers où le nouveau système LFS sera compilé et installé. Le Chapitre 3 explique quels paquets et correctifs ont besoin d'être téléchargés pour construire un système LFS et comment les stocker sur le nouveau système de fichiers. Le Chapitre 4 traite de la configuration pour un environnement de travail approprié. Lisez le Chapitre 4 avec attention, car il explique plusieurs problèmes importants que vous devez connaître avant de commencer à travailler sur le Chapitre 5 et les chapitres suivants.

Le Chapitre 5 explique l'installation de la chaîne d'outils initiale (binutils, gcc et glibc) avec une technique de compilation croisée qui isole les nouveaux outils du système hôte.

Le Chapitre 6 montre comment vous pouvez compiler les outils de base avec la chaîne de construction croisée tout juste construite.

Le Chapitre 7 entre ensuite dans un environnement « chroot » où nous utilisons les outils précédemment construits pour construire tous les outils nécessaires restants nécessaires à la construction et à la vérification du système final.

Cet effort consistant à isoler le nouveau système de la distribution hôte peut sembler excessif. Une explication technique complète est fournie dans Remarques techniques sur la chaîne de compilation.

Dans le Chapitre 8, le système LFS complet est construit. Un autre avantage fournit par l'environnement chroot est qu'il vous permet de continuer à utiliser le système hôte durant la construction de LFS. Vous pouvez continuer à utiliser votre ordinateur normalement en attendant la fin de la construction d'un paquet.

Pour terminer l'installation, la configuration de base du système est ajoutée dans le Chapitre 9, le noyau et le chargeur d'amorçage sont configurés dans le Chapitre 10. Le Chapitre 11 contient des informations sur la suite de l'expérience LFS après ce livre. Après avoir suivi les étapes de ce livre, l'ordinateur sera prêt à redémarrer dans le nouveau système LFS.

Ceci expose rapidement le processus. Des informations détaillées sur chaque étape sont traitées dans les chapitres suivants. Les éléments qui peuvent sembler compliqués seront clarifiés et tout prendra sens, alors que vous vous embarquez pour l'aventure LFS.

1.2. Nouveautés depuis la dernière version

Voici la liste des mises à jour de paquets opérées depuis la version précédente de LFS.

Mis à jour vers :

-
- Automake-1.18.1
- Bash-5.3
- Binutils-2.45
- Coreutils-9.7
- Diffutils-3.12
- E2fsprogs-1.47.3
- Expat-2.7.1
- Flit-Core-3.12.0
- Gawk-5.3.2
- GCC-15.2.0
- GDBM-1.26
- Gettext-0.26
- Glibc-2.42
- Gperf-3.3
- Grep-3.12
- Gzip-1.14
- Iana-Etc-20250807
- IPRoute2-6.16.0
- Jinja2-3.1.6
- Kbd-2.8.0
- Kmod-34.2
- Less-679
- LFS-Bootscripts-20250827
- Libcap-2.76
- Libelf de Elfutils-3.12
- Libffi-3.5.2
- Linux-6.16.1
- M4-1.4.20
- Man-DB-2.13.1
- Man-pages-6.15
- Meson-1.8.3
- MPFR-4.2.2
- Ncurses-6.5-20250809
- Ninja-1.13.1
- OpenSSL-3.5.2

- Patch-2.8
- Perl-5.42.0
- Pkgconf-2.5.1
- Python-3.13.7
- Readline-8.3
- Setuptools-80.9.0
- Shadow-4.18.0
- Sysklogd-2.7.2
- Systemd-257.8
- Tzdata-2025b
- Util-linux-2.41.1
- Vim-9.1.1629
- Wheel-0.46.1
- Xz-5.8.1

Ajoutés :

-
- Packaging-25.0
- coreutils-9.7-upstream_fix-1.patch

Supprimés :

-
- Check-0.15.2

1.3. Historique des modifications

Il s'agit de la version 12.4 du livre Linux From Scratch, datant du 01 septembre 2025. Si ce livre est daté de plus de six mois, une version plus récente et améliorée est probablement déjà disponible. Pour en avoir le cœur net, vérifiez la présence d'une nouvelle version sur l'un des miroirs via *https://www.linuxfromscratch.org/mirrors.html*.

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

Entrées dans l'historique des modifications :

- 01-09-2025
  - [bdubbs] — Publication de LFS-12.4.
- 27-08-2025
  - [bdubbs] - Mise à jour d'un cas particulier du script de démarrage réseau.
- 15-08-2025
  - [bdubbs] - Ajout d'un changement amont dans glibc pour résoudre une incompatibilité avec Valgrind. Corrige *#5778*.
  - [bdubbs] — Mise à jour vers iana-etc-20250807. Corrige *#5006*.
  - [bdubbs] — Mise à jour vers vim-9.1.1829. Corrige *#4500*.

- [bdubbs] — Mise à jour vers ncurses-6.5-20250809. Corrige *#5780*.
- [bdubbs] — Mise à jour vers Python-3.13.7 (mise à jour de sécurité). Corrige *#5779*.
- [bdubbs] — Mise à jour vers linux-6.16.1. Corrige *#5758*.
- [bdubbs] — Mise à jour vers iproute2-6.16.0. Corrige *#5773*.
- [bdubbs] — Mise à jour vers systemd-257.8. Corrige *#5751*.

- 08-08-2025
  - [bdubbs] — Mise à jour vers Python-3.13.6 (mise à jour de sécurité). Corrige *#5776*.
  - [bdubbs] — Mise à jour vers openssl-3.5.2. Corrige *#5775*.
  - [bdubbs] — Mise à jour vers libffi-3.5.2. Corrige *#5772*.
  - [bdubbs] — Mise à jour vers gcc-15.2.0. Corrige *#5777*.
- 05-08-2025
  - [renodr] — Correction de la CVE-2025-8194 dans Python. Corrige *#5774*.
- 01-08-2025
  - [bdubbs] — Mise à jour vers meson-1.8.3. Corrige *#5771*.
  - [bdubbs] — Mise à jour vers gdbm-1.26. Corrige *#5770*.
  - [bdubbs] — Mise à jour vers binutils-2.45. Corrige *#5766*.
  - [bdubbs] — Mise à jour vers gettext-0.26. Corrige *#5762*.
  - [bdubbs] — Mise à jour vers glibc-2.42. Corrige *#5765*.
  - [bdubbs] — Mise à jour vers man-pages-6.15. Corrige *#5763*.
- 15-07-2025
  - [bdubbs] — Mise à jour vers vim-9.1.1552 (mise à jour de sécurité). Corrige *#5760*.
  - [bdubbs] — Mise à jour vers readline-8.3. Corrige *#5755*.
  - [bdubbs] — Mise à jour vers perl-5.42.0. Corrige *#5756*.
  - [bdubbs] — Mise à jour vers openssl-3.5.1. Corrige *#5723*.
  - [bdubbs] — Mise à jour vers ninja-1.13.1. Corrige *#5759*.
  - [bdubbs] — Mise à jour vers linux-6.15.6. Corrige *#5757*.
  - [bdubbs] — Mise à jour vers gettext-0.25.1. Corrige *#5753*.
  - [bdubbs] — Mise à jour vers e2fsprogs-1.47.3. Corrige *#5758*.
  - [bdubbs] — Mise à jour vers bash-5.3. Corrige *#5754*.
- 01-07-2025
  - [bdubbs] — Mise à jour vers iana-etc-20250618. Corrige *#5006*.
  - [bdubbs] — Mise à jour vers vim-9.1.1497. Corrige *#4500*.
  - [bdubbs] — Mise à jour vers util-linux-2.41.1. Corrige *#5749*.
  - [bdubbs] — Mise à jour vers shadow-4.18.0. Corrige *#5750*.
  - [bdubbs] — Mise à jour vers pkgconf-2.5.1. Corrige *#5746*.
  - [bdubbs] — Mise à jour vers ninja-1.13.0. Corrige *#5745*.
  - [bdubbs] — Mise à jour vers linux-6.15.4. Corrige *#5748*.
  - [bdubbs] — Mise à jour vers less-679. Corrige *#5747*.

- [bdubbs] — Mise à jour vers automake-1.18.1. Corrige *#5752*.

- 15-06-2025
  - [bdubbs] — Mise à jour vers meson-1.8.2. Corrige *#5742*.
  - [bdubbs] — Mise à jour vers linux-6.15.2. Corrige *#5725*.
  - [bdubbs] — Mise à jour vers libffi-3.5.1. Corrige *#5741*.
  - [bdubbs] — Mise à jour vers iproute2-6.15.0. Corrige *#5732*.
  - [bdubbs] — Mise à jour vers Python-3.13.5. Corrige *#5743*.
- 04-06-2025
  - [bdubbs] — Mise à jour vers ncurses-6.5-20250531. Corrige *#5737*.
  - [bdubbs] — Mise à jour vers readline-8.3-rc2. Corrige *#5738*.
  - [bdubbs] — Mise à jour vers bash-5.3-rc2. Corrige *#5738*.
  - [bdubbs] — Mise à jour vers Python-3.13.4. Corrige *#6739*.
- 01-06-2025
  - [bdubbs] — Mise à jour vers iana-etc-20250519. Corrige *#5006*.
  - [bdubbs] — Mise à jour vers vim-9.1.1418. Corrige *#4500*.
  - [bdubbs] — Mise à jour vers kbd-2.8.0. Corrige *#5736*.
  - [bdubbs] — Mise à jour vers systemd-257.6. Corrige *#5674*.
  - [bdubbs] — Mise à jour vers setuptools-80.9.0. Corrige *#5728*.
  - [bdubbs] — Mise à jour vers meson-1.8.1. Corrige *#5731*.
  - [bdubbs] — Mise à jour vers automake-1.18. Corrige *#5734*.
  - [bdubbs] — Mise à jour des instructions de construction pour prendre en compte gcc-15 dans bc, expect, ncurses et gmp.
  - [bdubbs] — Mise à jour vers gcc-15.1.0. Corrige *#5707*.
  - [bdubbs] — Mise à jour vers less-678. Corrige *#5724*.
  - [bdubbs] — Mise à jour vers readline-8.3-rc1. Corrige *#5726*.
  - [bdubbs] — Mise à jour vers bash-5.3-rc1. Corrige *#5714*.
- 15-05-2025
  - [bdubbs] — Mise à jour vers setuptools-80.7.1. Corrige *#5715*.
  - [bdubbs] — Mise à jour vers man-pages-6.14. Corrige *#5720*.
  - [bdubbs] — Mise à jour vers man-db-2.13.1. Corrige *#5719*.
  - [bdubbs] — Mise à jour vers m4-1.4.20. Corrige *#5722*.
  - [bdubbs] — Mise à jour vers linux-6.14.6. Corrige *#5717*.
  - [bdubbs] — Mise à jour vers gettext-0.25. Corrige *#5718*.
- 01-05-2025
  - [bdubbs] — Mise à jour vers vim-9.1.1353. Corrige *#4500*.
  - [bdubbs] — Mise à jour vers setuptools-80.0.1. Corrige *#5710*.
  - [bdubbs] — Mise à jour vers packaging-25.0. Corrige *#5706*.
  - [bdubbs] — Mise à jour vers meson-1.8.0. Corrige *#5713*.

- [bdubbs] — Mise à jour vers linux-6.14.4. Corrige *#5709*.
- [bdubbs] — Mise à jour vers iana-etc-20250407. Corrige *#5006*.
- [bdubbs] — Mise à jour vers gperf-3.3. Corrige *#5708*.
- [bdubbs] — Mise à jour vers elfutils-0.193. Corrige *#5711*.

- 15-04-2025
  - [bdubbs] — Mise à jour vers libcap-2.76. Corrige *#5704*.
  - [bdubbs] — Mise à jour vers perl-5.40.2 (mise à jour de sécurité). Corrige *#5703*.
  - [bdubbs] — Ajout de packaging-24.2 (module Python). Requis pour wheel.
  - [bdubbs] — Mise à jour vers xz-5.8.1. Corrige *#5694*.
  - [bdubbs] — Mise à jour vers wheel-0.46.1 (module Python). Corrige *#5693*.
  - [bdubbs] — Mise à jour vers sysklogd-2.7.2. Corrige *#5690*.
  - [bdubbs] — Mise à jour vers Python3-3.13.3. Corrige *#5697*.
  - [bdubbs] — Mise à jour vers openssl-3.5.0. Corrige *#5701*.
  - [bdubbs] — Mise à jour vers meson-1.7.2. Corrige *#5691*.
  - [bdubbs] — Mise à jour vers linux-6.14.2. Corrige *#5680*.
  - [bdubbs] — Mise à jour vers libffi-3.4.8. Corrige *#5700*.
  - [bdubbs] — Mise à jour vers iproute2-6.14.0. Corrige *#5682*.
  - [bdubbs] — Mise à jour vers gzip-1.14. Corrige *#5699*.
  - [bdubbs] — Mise à jour vers grep-3.12. Corrige *#5702*.
  - [bdubbs] — Mise à jour vers gperf-3.2.1. Corrige *#5695*.
  - [bdubbs] — Mise à jour vers gawk-5.3.2. Corrige *#5692*.
  - [bdubbs] — Mise à jour vers diffutils-3.12. Corrige *#5696*.
  - [bdubbs] — Mise à jour vers coreutils-9.7. Corrige *#5698*.
- 01-04-2025
  - [bdubbs] — Mise à jour vers vim-9.1.1263. Corrige *#4500*.
  - [bdubbs] — Mise à jour vers iana-etc-20250328. Corrige *#5006*.
  - [bdubbs] — Mise à jour vers xz-5.8.0. Corrige *#5684*.
  - [bdubbs] — Mise à jour vers util-linux-2.41. Corrige *#5648*.
  - [bdubbs] — Mise à jour vers tzdata-2025b. Corrige *#5681*.
  - [bdubbs] — Mise à jour vers shadow-4.17.4. Corrige *#5678*.
  - [bdubbs] — Mise à jour vers setuptools-78.1.0. Corrige *#5676*.
  - [bdubbs] — Mise à jour vers patch-2.8. Corrige *#5689*.
  - [bdubbs] — Mise à jour vers mpfr-4.2.2. Corrige *#5677*.
  - [bdubbs] — Mise à jour vers kmod-34.2. Corrige *#5688*.
  - [bdubbs] — Mise à jour vers gdbm-1.25. Corrige *#5679*.
  - [bdubbs] — Mise à jour vers flit_core-3.12.0. Corrige *#5683*.
  - [bdubbs] — Mise à jour vers expat-2.7.1. Corrige *#5685*.
- 15-03-2025

- [bdubbs] — Mise à jour vers vim-9.1.1202. Corrige *#4500*.
- [bdubbs] — Mise à jour vers iana-etc-20250304. Corrige *#5006*.
- [bdubbs] — Mise à jour vers sysklogd-2.7.1. Corrige *#5668*.
- [bdubbs] — Mise à jour vers setuptools-76.0.0. Corrige *#5665*.
- [bdubbs] — Mise à jour vers pkgconf-2.4.3. Corrige *#5672*.
- [bdubbs] — Mise à jour vers man-pages-6.13. Corrige *#5673*.
- [bdubbs] — Mise à jour vers linux-6.13.7. Corrige *#5664*.
- [bdubbs] — Mise à jour vers libcap-2.75. Corrige *#5667*.
- [bdubbs] — Mise à jour vers kmod-34.1. Corrige *#5671*.
- [bdubbs] — Mise à jour vers jinja2-3.1.6. Corrige *#5670*.
- [bdubbs] — Mise à jour vers expat-2.7.0. Corrige *#5675*.

- 05-03-2025
  - [bdubbs] — Publication de LFS-12.3.

1.4. Ressources

1.4.1. FAQ

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

1.4.2. Listes de diffusion

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

Pour connaître les listes disponibles, les conditions d'abonnement, l'emplacement des archives et quelques autres informations, allez sur *http://fr.linuxfromscratch.org/aide*.

1.4.3. IRC

Plusieurs membres de la communauté LFS offrent leur aide sur IRC. Avant d'utiliser ce mode de communication, assurez-vous que la réponse à votre question ne se trouve pas déjà dans la FAQ LFS ou dans les archives des listes de diffusion. Vous trouverez le réseau IRC à l'adresse `irc.libera.chat`. Le canal d'entre-aide en français se nomme #lfs-fr.

1.4.4. Sites miroirs

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

1.4.5. Contacts

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

1.5. Aide

Note

Si vous avez un problème pour construire l'un des paquets avec les instructions de LFS, nous vous déconseillons fortement d'envoyer le problème directement sur les canaux de support en amont avant d'en parler sur l'un des canaux de support LFS répertoriés dans Section 1.4, « Ressources. » C'est souvent peu efficace car les développeurs en amont ne sont pas souvent au courant des procédures de construction de LFS. Même si vous avez vraiment rencontré un problème amont, la communauté LFS peut quand même vous aider à isoler l'information demandée en amont et à rédiger un rapport correct.

Si vous devez poser une question directement via un canal de support en amont, vous devez au moins prendre en compte que la plupart des projets ont un canal de support séparé du gestionnaire de bogues. Les rapports de « boque » qui posent des questions sont considérés comme invalides et peuvent embêter les développeurs amont de ces projets.

Si vous rencontrez une erreur ou si vous vous posez une question en travaillant avec ce livre, merci de vérifier la FAQ sur *https://fr.linuxfromscratch.org/faq#generalfaq*. Vous y trouverez souvent la réponse à vos questions. Dans le cas contraire, essayez de trouver la source du problème. Le guide suivant vous donnera quelques conseils pour cela : *https://fr.linuxfromscratch.org/view/astuces/errors.txt*.

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

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

1.5.1. Éléments à mentionner

En plus d'une brève explication du problème que vous rencontrez, voici les éléments essentiels à inclure dans votre demande d'aide :

- La version du livre que vous utilisez (dans ce cas, 12.4)
- La distribution hôte (et sa version) que vous utilisez pour créer LFS
- La sortie du script Prérequis du système hôte
- Le paquet ou la section où le problème a été rencontré
- Le message d'erreur exact ou une description claire du problème
- Indiquez si vous avez dévié du livre

Note

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

1.5.2. Problèmes du script configure

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

1.5.3. Problèmes de compilation

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

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

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

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

Cette information n'est pas suffisante pour diagnostiquer correctement le problème, car elle indique seulement que quelque chose s'est mal passé, pas *ce qui* s'est mal passé. C'est la section entière, comme dans l'exemple plus haut, qui devrait être copiée, car la commande exécutée et tout message d'erreur associé sont inclus.

Un excellent article sur les demandes d'aide sur Internet est disponible en ligne sur *http://www.gnurou.org/writing/smartquestionsfr*. Lisez et suivez les astuces de ce document pour accroître vos chances d'obtenir l'aide dont vous avez besoin.

Partie II. Préparation à la construction

Chapitre 2. Préparation du système hôte

2.1. Introduction

Dans ce chapitre, la présence des outils du système hôte nécessaires à la construction de LFS est vérifiée. Si besoin, ils sont installés avant de préparer la partition qui contiendra le système LFS. Nous créerons la partition elle-même, nous lui ajouterons un système de fichiers et nous la monterons.

2.2. Prérequis du système hôte

2.2.1. Matériel

Les éditeurs de LFS recommandent d'avoir un CPU avec au moins 4 cœurs et un système avec au moins 8 Go de mémoire. Les systèmes plus vieux qui n'atteignent pas ces prérequis fonctionneront quand même mais le temps de construction des paquets sera bien plus long que ce qui est documenté.

2.2.2. Logiciel

Votre système hôte doit contenir les logiciels suivants dans la version minimum indiquée. Cela ne devrait pas être un problème pour la plupart des distributions modernes de Linux. Notez également que certaines distributions placeront les en-têtes des logiciels dans des paquets distincts, ayant souvent la forme suivante : `<nom-du-paquet>-devel` ou `<nom-du-paquet>-dev`. Assurez-vous de les installer si votre distribution les fournit.

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

- **Bash-3.2** (/bin/sh doit être un lien symbolique ou matériel vers bash)
- **Binutils-2.13.1** (les versions ultérieures à 2.45 ne sont pas recommandées car elles n'ont pas été testées)
- **Bison-2.7** (/usr/bin/yacc doit être un lien vers bison ou un petit script qui exécute bison)
- **Coreutils-8.1**
- **Diffutils-2.8.1**
- **Findutils-4.2.31**
- **Gawk-4.0.1** (/usr/bin/awk doit être un lien vers gawk)
- **GCC-5.4**, y compris le compilateur C++, **g++** (les versions ultérieures à 15.2.0 ne sont pas recommandées, car elles n'ont pas été testées). Les bibliothèques standards C et C++ (avec les headers) doivent également être présentes pour que le compilateur C++ puisse construire les programmes hébergés
- **Grep-2.5.1a**
- **Gzip-1.3.12**
- **Noyau Linux-5.4**

  La raison pour laquelle cette version du noyau est requise est que nous spécifions cette version lors de la construction de glibc dans Chapitre 5 et Chapitre 8, pour que les contournements pour les anciens noyaux ne soient pas activés et que la glibc compilée soit un peu plus rapide et plus petite. En décembre 2024, 5.4 est la plus vieille version du noyau prise en charge par les développeurs du noyau. Certaines versions du noyau plus vieilles que 5.4 peuvent toujours être prises en charge par des équipes tierces, mais elles ne sont pas considérées comme des versions officielles en amont. Consultez *https://kernel.org/category/releases.html* pour plus de détails.

  Si le noyau hôte est plus ancien que le 5.4, vous devez le remplacer avec une version plus récente. Il existe deux méthodes de remplacement. Dans un premier temps, vérifiez si votre distribution Linux fournit un paquet pour le noyau 5.4 ou pour un noyau plus récent. Si c'est le cas, vous pouvez l'installer. Si votre distribution ne propose pas de paquet correspondant pour le noyau, ou si vous préférez ne pas l'installer, vous pouvez compiler le noyau vous-même. Les instructions pour la compilation du noyau et la configuration du chargeur d’amorçage (en supposant que le système hôte utilise GRUB) sont disponibles au Chapitre 10.

Nous avons besoin que le noyau hôte prenne en charge les pseudo-terminaux UNIX 98 (PTY). Cela devrait être activé sur toutes les distributions de bureau et de serveur qui utilisent Linux 5.4 ou supérieur. Si vous construisez sur un noyau hôte personnalisé, assurez-vous que l'option `CONFIG_UNIX98_PTYS` est indiquée à `y` dans la configuration du noyau.

- **M4-1.4.10**
- **Make-4.0**
- **Patch-2.5.4**
- **Perl-5.8.8**
- **Python-3.4**
- **Sed-4.1.5**
- **Tar-1.22**
- **Texinfo-5.0**
- **Xz-5.0.0**

Important

Remarque : les liens symboliques mentionnés ci-dessus sont nécessaires pour construire un système LFS en utilisant les instructions contenues dans ce livre. Les liens symboliques qui renvoient vers d'autres logiciels (comme dash, mawk, etc.) peuvent fonctionner, mais ils n'ont pas été testés, ou bien ne sont pas pris en charge par l'équipe de développement LFS. Il est possible que ces liens nécessitent soit de vous écarter des instructions du livre, soit d'apporter des correctifs supplémentaires à certains paquets.

Pour vérifier que votre système hôte possède toutes les versions nécessaires et peut compiler des programmes, exécutez la commande suivante :

```
cat > version-check.sh << "EOF"
#!/bin/bash
# Un script qui liste les numéro de version des outils de développement critiques

# Si vous avez des outils installés dans d'autres répertoires, ajustez PATH ici ET
# dans ~lfs/.bashrc (section 4.4) également.

LC_ALL=C
PATH=/usr/bin:/bin

bail() { echo "FATAL : $1"; exit 1; }
grep --version > /dev/null 2> /dev/null || bail "grep ne fonctionne pas"
sed '' /dev/null || bail "sed ne fonctionne pas"
sort   /dev/null || bail "sort ne fonctionne pas"

ver_check()
{
   if#! type -p $2 &>/dev/null
   then
     echo "ERREUR : $2 ($1) introuvable"; return 1;
   fi
   v=$($2 --version 2>&1 | grep -E -o '[0-9]+\.[0-9\.]+[a-z]*' | head -n1)
   if printf '%s\n' $3 $v | sort --version-sort --check &>/dev/null
   then
     printf "OK :     %-9s %-6s >= $3\n" "$1" "$v"; return 0;
   else
     printf "ERREUR : %-9s est TROP VIEUX (version $3 ou supérieure requise)\n" "$1";
     return 1;
   fi
}

ver_kernel()
{
   kver=$(uname -r | grep -E -o '^[0-9\.]+')
   if printf '%s\n' $1 $kver | sort --version-sort --check &>/dev/null
```

```
   then
     printf "OK :     noyau Linux $kver >= $1\n"; return 0;
   else
     printf "ERREUR : noyau Linux ($kver) est TROP VIEUX (version $1 ou supérieure requise)\n" "$kver";
     return 1;
   fi
}

# Coreutils en premier car --version-sort a besoin de Coreutils >= 7.0
ver_check Coreutils      sort     8.1 || bail "Coreutils trop vieux, arrêt"
ver_check Bash           bash     3.2
ver_check Binutils       ld       2.13.1
ver_check Bison          bison    2.7
ver_check Diffutils      diff     2.8.1
ver_check Findutils      find     4.2.31
ver_check Gawk           gawk     4.0.1
ver_check GCC            gcc      5.4
ver_check "GCC (C++)"    g++      5.4
ver_check Grep           grep     2.5.1a
ver_check Gzip           gzip     1.3.12
ver_check M4             m4       1.4.10
ver_check Make           make     4.0
ver_check Patch          patch    2.5.4
ver_check Perl           perl     5.8.8
ver_check Python         python3  3.4
ver_check Sed            sed      4.1.5
ver_check Tar            tar      1.22
ver_check Texinfo        texi2any 5.0
ver_check Xz             xz       5.0.0
ver_kernel 5.4

if mount | grep -q 'devpts on /dev/pts' && [ -e /dev/ptmx ]
then echo "OK :     le noyau Linux prend en charge les PTY UNIX 98";
else echo "ERREUR : le noyau Linux ne prend PAS en charge les PTY UNIX 98"; fi

alias_check() {
   if $1 --version 2>&1 | grep -qi $2
   then printf "OK :     %-4s est $2\n" "$1";
   else printf "ERREUR : %-4s n'est PAS $2\n" "$1"; fi
}
echo "Alias :"
alias_check awk GNU
alias_check yacc Bison
alias_check sh Bash

echo "Vérification du compilateur :"
if printf "int main(){}" | g++ -x c++ -
then echo "OK :     g++ fonctionne";
else echo "ERREUR : g++ ne fonctionne PAS"; fi
rm -f a.out

if [ "$(nproc)" = "" ]; then
   echo "ERREUR : nproc n'est pas disponible ou a produit une sortie vide"
else
   echo "OK : nproc rapporte $(nproc) cœurs logiques disponibles"
fi
EOF

bash version-check.sh
```

2.3. Les étapes de la construction de LFS

LFS est conçu pour être construit en une seule session. En d'autres termes, les instructions supposent que le système ne sera pas éteint pendant la construction. Cela ne signifie pas que le système doit être construit d'une traite. Le problème, c'est que certaines procédures devront être relancées après un redémarrage si vous reprenez la construction de LFS à différentes étapes.

2.3.1. Chapitres 1 à 4

Ces chapitres exécutent des commandes sur le système hôte. En cas de redémarrage, vérifiez une chose :

- Les procédures effectuées en tant qu'utilisateur `root` après la Section 2.4 ont besoin que la variable d'environnement LFS soit définie *POUR L'UTILISATEUR ROOT*.

2.3.2. Chapitres 5–6

- La partition /mnt/lfs doit être montée.
- Ces deux chapitres *doivent* être effectués en tant qu'utilisateur `lfs`. Vous devez exécuter **su - lfs** avant d'effectuer quoi que ce soit dans ces chapitres. Si vous ne le faites pas, vous risquez d'installer des paquets sur l'hôte et éventuellement de le rendre inutilisable.
- Les procédures de Instructions générales de compilation sont cruciales. Si vous avez le moindre doute sur l'installation correcte d'un paquet, assurez-vous d'avoir supprimé toute archive décompressée, extrayez de nouveau les fichiers du paquet et suivez toutes les instructions de cette section.

2.3.3. Chapitres 7–10

- La partition /mnt/lfs doit être montée.
- Certaines opérations, de « Préparer les systèmes de fichiers virtuels du noyau » à « Entrer dans l'environnement chroot » doivent être effectuées en tant qu'utilisateur `root`, avec la variable d'environnement configurée pour l'utilisateur `root`.
- En entrant dans l'environnement chroot, la variable d'environnement LFS doit être définie pour l'utilisateur `root`. La variable LFS n'est plus utilisée après l'entrée dans l'environnement chroot.
- Les systèmes de fichiers virtuels doivent être montés. Ceci peut se faire avant ou après être entré dans l'environnement chroot en changeant de terminal dans le système hôte et, en tant que `root`, en exécutant les commandes de la Section 7.3.1, « Monter et alimenter /dev » et de la Section 7.3.2, « Monter les systèmes de fichiers virtuels du noyau. »

2.4. Création d'une nouvelle partition

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

Un système minimal requiert une partition d'environ 10 Go. Cet espace est suffisant pour conserver toutes les archives tar des sources et pour compiler tous les paquets. Néanmoins, si le système LFS a pour but d'être un système Linux primaire, des logiciels additionnels seront probablement installés et nécessiteront de l'espace supplémentaire. Une partition de 30 Go devrait suffire à garantir l'espace nécessaire à cette opération. Le système LFS en lui-même ne prendra pas tout cet espace. Une grande partie de cet espace servira à prévoir suffisamment d'espace temporaire libre ainsi qu'à accueillir des fonctionnalités supplémentaires une fois LFS terminé. De plus, la compilation des paquets peut demander beaucoup d'espace disque, qui sera récupéré après l'installation du paquet.

Parce qu'il n'y a pas toujours assez de mémoire vive (RAM) disponible pour les processus de compilation, il est conseillé d'utiliser une petite partition comme espace d'échange `swap`. Le noyau utilise cet espace pour stocker des données rarement utilisées et pour laisser plus de mémoire aux processus actifs. Il se peut que la partition `swap` pour un système LFS soit la même que celle utilisée par le système hôte, auquel cas il n'est pas nécessaire de créer une autre partition si votre système hôte possède déjà cette configuration.

Lancez un programme de partitionnement de disques, tel que **cfdisk** ou **fdisk** avec une option en ligne de commande nommant le disque dur sur lequel la nouvelle partition sera créée. Par exemple, on utilisera `/dev/sda` pour le disque primaire. Créez une partition Linux native et, si nécessaire, une partition `swap`. Référez-vous au manuel de *cfdisk(8)* ou de *fdisk(8)* si vous ne savez pas encore utiliser les programmes.

Note

Pour les utilisateurs expérimentés, il est possible d'utiliser d'autres méthodes de partitionnement. Le nouveau système LFS peut être installé sur un logiciel *RAID* ou sur un volume logique *LVM*. Par contre, certaines options exigent un *initramfs*, qui concerne un sujet plus avancé. Ces méthodes de partitionnement ne sont pas recommandées pour les utilisateurs novices de LFS.

Retenez bien la désignation de la nouvelle partition (par exemple `sda5`). Elle sera référencée dans ce livre comme partition LFS. Gardez également en mémoire la désignation de la partition `swap`. Ces noms vous seront utiles pour le fichier `/etc/fstab`.

2.4.1. Autres problèmes liés au partitionnement

Les demandes de conseils par rapport au partitionnement du système sont souvent postées sur les listes de diffusion LFS. Ce sujet est très subjectif. Par défaut, la plupart des distributions utilisent le disque entier, sauf une petite partie réservée à la partition d'échange. Cette solution n'est pas optimale pour LFS, et ce pour plusieurs raisons. Cela réduit la flexibilité, complique le partage de données entre plusieurs distributions ou constructions LFS et allonge le temps de sauvegarde. Cela peut également faire perdre de l'espace disque à cause d’une allocation inefficace des structures de fichiers.

2.4.1.1. La partition racine

Une partition racine LFS (à ne pas confondre avec le répertoire `/root`), de 20 Go est un bon compromis pour la plupart des systèmes. Elle fournit assez d’espace pour construire LFS et une grande partie de BLFS, tout en étant assez petit pour que plusieurs partitions puissent être créées facilement à des fins expérimentales.

2.4.1.2. La partition d'échange (swap)

La plupart des distributions créent automatiquement une partition d'échange. En général, la taille recommandée pour une partition d'échange est à peu près deux fois supérieure à la taille de la RAM physique, bien que ce soit rarement nécessaire. Si votre espace disque est limité, laissez la partition d'échange à 2 Go et surveillez l'utilisation de la mémoire d'échange sur le disque.

Si vous voulez utiliser le mode hibernation (veille sur disque) de Linux, il transfère le contenu de la RAM vers la partition d'échange avant d'éteindre la machine. Dans ce cas, la partition d'échange doit être au moins aussi grande que la RAM installée sur le système.

L'utilisation de la mémoire d'échange n'est jamais une bonne chose. Avec un disque dur mécanique, vous pouvez déterminer si un système utilise la mémoire d'échange simplement en écoutant l'activité du disque et en observant la façon dont le système réagit aux commandes. Vous ne pourrez pas l'entendre utiliser l'espace d'échange avec un disque SSD, mais vous pouvez savoir combien d'espace d'échange est utilisé avec les programmes **top** et **free**. Si possible, vous devez éviter d'utiliser une partition d'échange sur un disque SSD. Lorsque la mémoire d’échange est

utilisée, vous devez en premier lieu vérifier si l'une des commandes n'est pas insensée, comme par exemple l'édition d'un fichier de 5 Go. Si l'utilisation de la mémoire d'échange devient un phénomène habituel, la meilleure solution est d'ajouter de la mémoire RAM à votre système.

2.4.1.3. La partition Bios de Grub

Si le *disque de démarrage* est partitionné avec une table de partition GUID (GPT), alors une petite partition de l'ordre d'1 Mo doit être créée, si elle n'existe pas déjà. Cette partition n'est pas formatée, mais doit être disponible pour que GRUB l'utilise pendant l'installation du chargeur d'amorçage. Cette partition sera normalement intitulée « *BIOS Boot* » si vous utilisez **fdisk** ou aura le code *EF02* avec **gdisk**.

Note

La partition Bios de Grub doit se trouver sur le disque que le BIOS utilise pour démarrer le système. Il ne s'agit pas nécessairement du même disque que celui sur lequel la partition racine de LFS est installée. Les disques d'un système peuvent utiliser des types de tables de partitionnement différents. Seul le type de table de partitionnement du disque de démarrage détermine si cette partition est nécessaire ou non.

2.4.1.4. Partitions de commodité

Il existe d'autres partitions qui ne sont pas indispensables, mais que vous devez envisager lorsque vous aménagez un disque dur. La liste suivante n'est pas exhaustive mais a été conçue pour vous guider.

- /boot – Fortement recommandée. Utilisez cette partition pour stocker les noyaux et d'autres informations de démarrage. Pour limiter les risques de problèmes de démarrage avec les disques volumineux, faites-en la première partition physique sur votre premier disque dur. Une taille de partition de 200 mégaoctets est parfaitement adaptée.
- /boot/efi – La partition EFI système, qui est requise pour démarrer le système avec UEFI. Consultez *la page BLFS* pour plus d'informations.
- /home – Fortement recommandée. Partage votre répertoire home et vos paramètres utilisateur entre plusieurs distributions ou constructions de LFS. La taille est généralement assez importante et dépend de l'espace disque disponible.
- /usr – Dans LFS, `/bin`, `/lib` et `/sbin` sont des liens symboliques vers leurs équivalents dans `/usr`. Le dossier `/usr` contient tous les binaires nécessaires à l'exécution du système. Dans LFS, une partition séparée pour `/usr` n'est généralement pas requise. Si vous souhaitez quand même en créer une, vous devrez faire en sorte qu'elle soit assez grande pour contenir tous les programmes et bibliothèques du système. La partition racine de cette configuration peut être très petite (par exemple seulement un gigaoctet). Elle est donc parfaitement adaptée à un client léger ou une station de travail sans disque (où `/usr` est monté depuis un serveur distant). Cependant, il faut savoir que vous aurez besoin d'un initramfs (dont la construction n'est pas évoquée dans LFS) pour démarrer le système avec une partition `/usr` séparée.
- /opt – Ce répertoire est surtout utile pour BLFS, où vous pouvez installer plusieurs versions de paquets volumineux tels que KDE ou Texlive sans incorporer les fichiers dans la hiérarchie /usr. Si vous l'utilisez, un espace de 5 à 10 gigaoctets est généralement adapté.
- /tmp – On utilise rarement une partition /tmp séparée, mais cela peut être utile si vous configurez un client léger. Si vous l'utilisez, cette partition ne prendra en général pas plus de 2 Go d'espace. Si votre mémoire RAM est suffisante, vous pouvez monter un `tmpfs` sur le /tmp pour accéder plus rapidement aux fichiers temporaires.
- /usr/src – Cette partition est très utile pour assurer un emplacement de stockage des fichiers source de BLFS et les partager entre les constructions LFS. Vous pouvez aussi l'utiliser comme un espace de construction des paquets BLFS. Une partition entre 30 et 50 gigaoctets offre suffisamment d'espace.

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

2.5. Création d'un système de fichiers sur la partition

Une partition est un ensemble de secteurs sur un lecteur de disque. Cet ensemble est délimité par des bornes placées dans un tableau de partitions. Avant que le système d'exploitation ne puisse utiliser une partition pour stocker des fichiers, il faut d'abord formater la partition. Elle pourra contenir un système de fichiers constitué principalement d'une étiquette interne, d'un répertoire de blocs, de blocs de données et d'un schéma d'indexation qui permettront de localiser un fichier précis sur demande. Le système de fichiers aide aussi l'OS à garder une trace de l'espace libre sur la partition, stocker les secteurs nécessaires quand un nouveau fichier est créé ou qu'un fichier existant est étendu et recycler les segments de données libres qui sont créés lorsque des fichiers sont supprimés. Le système de fichiers peut aussi servir de support en cas de redondance des données et d'élimination d'erreurs.

Maintenant qu'une partition vierge est prête, le système de fichiers peut être créé. LFS peut utiliser n'importe quel système de fichiers reconnu par le noyau Linux, mais les types ext3 et ext4 sont les plus communs. Le choix d'un système de fichiers peut être complexe et dépend des caractéristiques des fichiers et de la taille de la partition. Par exemple :

ext2
: convient aux petites partitions rarement mises à jour, telles que /boot.

ext3
: est une mise à jour d'ext2 qui comprend un journal aidant à récupérer l'état de la partition en cas d'arrêt brutal. Ce type est communément employé dans une perspective généraliste.

ext4
: est la dernière version de type ext des systèmes de fichiers. Ce type offre de nouvelles possibilités, notamment l'horodatage à la nanoseconde, la création et l'utilisation de très gros fichiers (16 To), ainsi que des améliorations de vitesse.

D'autres systèmes de fichiers comme FAT32, NTFS, JFS et XFS servent à des fins plus spécifiques. Vous pouvez trouver plus d'informations sur ces systèmes de fichiers sur *https://fr.wikipedia.org/wiki/Liste_des_syst%C3%A8mes_de_fichiers*.

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

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

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

Si vous utilisez une partition `swap` déjà existante, il n'est pas nécessaire de la formater. En cas de création d'une nouvelle partition `swap`, initialisez-la à l'aide de la commande suivante :

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

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

2.6. Définition de la variable $LFS et du Umask

Tout au long de ce livre, la variable d'environnement `LFS` est mentionnée à plusieurs reprises. Assurez-vous de toujours définir cette variable pendant le processus de construction de LFS. Cette variable doit être associée au nom du répertoire où vous construirez votre système LFS : nous utiliserons `/mnt/lfs` comme exemple mais le choix du répertoire vous appartient. Si vous construisez LFS sur une partition séparée, le répertoire défini sera le point de montage de la partition. Choisissez un répertoire et définissez la variable avec la commande suivante :

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

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

Maintenant, configurez le masque de création de fichier (umask) à `022` si la distribution hôte utilise une valeur par défaut différente :

```
umask 022
```

Configurer l'umask à 022 s'assure que les fichiers et les répertoires nouvellement créés ne sont inscriptibles que pour leur propriétaire, mais restent lisibles et trouvables (seulement pour les répertoires) par n'importe qui (en supposant que les modes par défaut sont utilisés par l'appel système *open(2)*, les nouveau fichiers obtiendront le mode de permission 644 et les répertoires le mode 755). Une valeur par défaut trop permissive peut laisser des trous de sécurité dans le système LFS, et une valeur par défaut trop restrictive peut causer des problèmes étranges lors de la construction ou de l'utilisation du système LFS.

Attention

N'oubliez pas de vérifier que la variable `LFS` est définie et que l'umask est bien `022` à chaque fois que vous quittez et revenez dans l'environnement de travail (lorsque vous exécutez, par exemple, **su** en `root` ou un autre utilisateur). Vérifiez que la variable `LFS` est définie correctement avec la commande suivante :

```
echo $LFS
```

Assurez-vous que la sortie affiche le chemin du répertoire dans lequel vous construisez votre système LFS, qui est `/mnt/lfs` si vous avez suivi l'exemple fourni.

Vérifiez que l'umask est configuré correctement avec :

```
umask
```

La sortie peut être `0022` ou `022` (le nombre de zéros initiaux dépend de la distribution hôte).

Si la sortie d'une de ces deux commandes est incorrecte, utilisez la commande donnée plus haut pour configurer `$LFS` au bon répertoire et configurer l'umask à `022`.

Note

Une manière de vous assurer que la variable `LFS` est l'umask sont toujours définis correctement est d'éditer le fichier `.bash_profile` à la fois dans votre répertoire personnel et dans le fichier `/root/.bash_profile` et d'y saisir les commandes **export** et **umask** mentionnées plus haut. De plus, pour tous les utilisateurs ayant besoin de la variable `LFS`, le shell indiqué dans le fichier `/etc/passwd` doit être « bash » afin de s'assurer que le fichier `.bash_profile` soit inclus dans le processus de connexion.

Une autre chose à prendre en compte est la méthode que vous utilisez pour vous connecter au système hôte. Si vous vous connectez via un gestionnaire d'affichage graphique, le fichier `.bash_profile` de l'utilisateur n'est normalement pas utilisé lorsque le gestionnaire lance un terminal virtuel. Dans ce cas, ajoutez les commandes au fichier `.bashrc` à la fois pour l'utilisateur et pour `root`. En plus, certaines distributions ont des instructions qui empêchent le chargement de `.bashrc` dans une invocation non interactive de bash. Assurez-vous d'ajouter les commandes avant le test pour l'utilisation non interactive si c'est le cas.

2.7. Montage de la nouvelle partition

Maintenant qu'un système de fichiers a été créé, la partition doit être montée afin que le système hôte puisse y accéder. Dans ce livre, on suppose que le système de fichiers est monté sur le répertoire spécifié par la variable d'environnement `LFS`, comme décrit dans la section précédente.

On ne peut pas « monter une partition » à proprement parler. On monte le *système de fichiers* intégré à cette partition. Mais comme une partition seule ne peut pas contenir plus d'un système de fichiers, la partition et le système de fichiers associé sont souvent considérés comme une seule et même entité.

Créez le point de montage et montez le système de fichiers LFS en exécutant la commande suivante :

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

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

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

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

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

Configurez le propriétaire et les permissions du répertoire `$LFS` (c'est-à-dire le répertoire racine du nouveau système de fichiers créé par le système LFS) à `root` et `755` si la distribution hôte a été configurée pour utiliser une valeur par défaut différente dans **mkfs** :

```
chown root:root $LFS
chmod 755 $LFS
```

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

Avertissement

Les instructions ci-dessus supposent que vous ne redémarrerez pas votre ordinateur pendant le processus LFS. Si vous éteignez votre système, vous devrez soit remonter la partition LFS à chaque redémarrage de la construction, soit modifier le fichier `/etc/fstab` de votre système hôte pour qu'il soit remonté automatiquement au démarrage. Par exemple, vous pouvez ajouter cette ligne à votre fichier `/etc/fstab` :

```
/dev/<xxx>  /mnt/lfs ext4   defaults      1     1
```

Si vous utilisez des partitions facultatives supplémentaires, assurez-vous de les ajouter aussi.

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

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

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

Maintenant que votre nouvelle partition LFS est prête à être utilisée, vous pouvez télécharger les paquets.

Chapitre 3. Paquets et correctifs

3.1. Introduction

Ce chapitre inclut une liste de paquets devant être téléchargés pour construire un système Linux de base. Les numéros de versions affichés correspondent aux versions des logiciels qui sont connues pour fonctionner, et ce livre est basé sur leur utilisation. Nous déconseillons fortement l'utilisation de versions différentes. En effet, les commandes de construction d'une version pourraient ne pas fonctionner sur une version différente, à moins que cette version différente ne soit mentionné dans un errata ou une information de sécurité. Les versions plus récentes pourraient aussi avoir des problèmes nécessitant des contournements. Ces derniers seront développés et stabilisés dans la version de développement du livre.

Pour certains paquets, l'archive de version et l'instantané du dépôt (Git ou SVN) pour cette version peuvent être publiés avec des noms de fichiers similaires, voire identiques. Cependant, une archive de version publiée peut contenir des fichiers essentiels bien que non présents dans le dépôt (par exemple, un script **configure** généré par **autoconf**) en plus du contenu de l'instantané du dépôt. Le livre utilise les archives de version chaque fois que cela est possible. Utiliser des instantanés de dépôt au lieu des archives de version spécifiées dans le livre causera des problèmes.

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

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

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

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

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

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

Il y a plusieurs moyen d'obtenir tous les paquets nécessaires et les correctifs requis pour construire LFS :

- Vous pouvez télécharger les fichiers individuellement comme décrit dans les deux prochaines sections.
- Pour les versions stables du livre, une archive de tous les fichiers nécessaires est disponible au téléchargement sur l'un des miroirs de LFS listés sur *https://www.linuxfromscratch.org/mirrors.html#files*.
- Vous pouvez télécharger les fichiers avec **wget** et un wget-list comme décrit ci-dessous.

Pour télécharger tous les paquets et les correctifs en utilisant *wget-list* comme entrée pour la commande **wget**, utilisez :

```
wget --input-file=wget-list-sysv --continue --directory-prefix=$LFS/sources
```

En outre, à partir de LFS-7.0, un fichier séparé, *md5sums*, peut être utilisé pour vérifier que tous les paquets sont disponibles avant de continuer. Mettez ce fichier dans `$LFS/sources` et lancez :

```
pushd $LFS/sources
  md5sum -c md5sums
popd
```

Vous pouvez utiliser cette vérification après avoir récupéré les fichiers nécessaires avec une des méthodes proposées plus haut.

Si vous téléchargez les paquets et les correctifs en tant qu'utilisateur non-`root`, ces fichiers appartiendront à l'utilisateur. Le système de fichiers identifie le propriétaire par son UID et l'UID d'un utilisateur normal dans la distribution hôte n'est pas attribué dans LFS. Dans ce cas, le propriétaire par défaut des fichiers sera un UID sans nom dans le système LFS final. Si vous n'attribuez pas le même UID à votre utilisateur dans le système LFS, changez dès maintenant les propriétaires de ces fichiers à `root` afin d'éviter le problème suivant :

```
chown root:root $LFS/sources/*
```

3.2. Tous les paquets

Note

Lisez les *annonces de sécurité* avant de télécharger les paquets pour découvrir s'il y a de nouvelles versions de paquets qui pourraient être utilisées pour éviter des problèmes de sécurité.

Les développeurs en amont peuvent supprimer les anciennes versions surtout si elles contiennent des vulnérabilités. Si une URL ci-dessous est injoignable, vous pouvez lire les annonces de sécurité avant pour savoir si vous devriez utiliser une version plus récente (avec la vulnérabilité corrigée). Sinon, essayez de télécharger le paquets supprimé depuis un miroir. Bien qu'il soit possible de télécharger une ancienne version à partir d'un miroir même si la version a été supprimée à cause d'une vulnérabilité, nous ne vous recommandons pas d'utiliser une version connue pour être vulnérable pour construire votre système.

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

• Acl (2.3.2) — 363 Ko :
Page d'accueil : *https://savannah.nongnu.org/projects/acl*
Téléchargement : *https://download.savannah.gnu.org/releases/acl/acl-2.3.2.tar.xz*
Somme de contrôle MD5 : `590765dee95907dbc3c856f7255bd669`

• Attr (2.5.2) — 484 Ko :
Page d'accueil : *https://savannah.nongnu.org/projects/attr*
Téléchargement : *https://download.savannah.gnu.org/releases/attr/attr-2.5.2.tar.gz*
Somme de contrôle MD5 : `227043ec2f6ca03c0948df5517f9c927`

• Autoconf (2.72) — 1 360 Ko :
Page d'accueil : *https://www.gnu.org/software/autoconf/*
Téléchargement : *https://ftp.gnu.org/gnu/autoconf/autoconf-2.72.tar.xz*
Somme de contrôle MD5 : `1be79f7106ab6767f18391c5e22be701`

• Automake (1.18.1) — 1 614 Ko :
Page d'accueil : *https://www.gnu.org/software/automake/*
Téléchargement : *https://ftp.gnu.org/gnu/automake/automake-1.18.1.tar.xz*
Somme de contrôle MD5 : `cea31dbf1120f890cbf2a3032cfb9a68`

• Bash (5.3) — 11 089 Ko :
Page d'accueil : *https://www.gnu.org/software/bash/*
Téléchargement : *https://ftp.gnu.org/gnu/bash/bash-5.3.tar.gz*
Somme de contrôle MD5 : `977c8c0c5ae6309191e7768e28ebc951`

• Bc (7.0.3) — 464 Ko :
Page d'accueil : *https://github.com/gavinhoward*
Téléchargement : *https://github.com/gavinhoward/bc/releases/download/7.0.3/bc-7.0.3.tar.xz*
Somme de contrôle MD5 : `ad4db5a0eb4fdbb3f6813be4b6b3da74`

• Binutils (2.45) — 27 216 Ko :
Page d'accueil : *https://www.gnu.org/software/binutils/*
Téléchargement : *https://sourceware.org/pub/binutils/releases/binutils-2.45.tar.xz*
Somme de contrôle MD5 : `dee5b4267e0305a99a3c9d6131f45759`

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

• Bzip2 (1.0.8) — 792 Ko :
Téléchargement : *https://www.sourceware.org/pub/bzip2/bzip2-1.0.8.tar.gz*
Somme de contrôle MD5 : `67e051268d0c475ea773822f7500d0e5`

• Coreutils (9.7) — 6 015 Ko :
Page d'accueil : *https://www.gnu.org/software/coreutils/*
Téléchargement : *https://ftp.gnu.org/gnu/coreutils/coreutils-9.7.tar.xz*
Somme de contrôle MD5 : `6b7285faf7d5eb91592bdd689270d3f1`

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

• Diffutils (3.12) — 1 894 Ko :
Page d'accueil : *https://www.gnu.org/software/diffutils/*
Téléchargement : *https://ftp.gnu.org/gnu/diffutils/diffutils-3.12.tar.xz*
Somme de contrôle MD5 : `d1b18b20868fb561f77861cd90b05de4`

• E2fsprogs (1.47.3) — 9 851 Ko :
Page d'accueil : *https://e2fsprogs.sourceforge.net/*
Téléchargement : *https://downloads.sourceforge.net/project/e2fsprogs/e2fsprogs/v1.47.3/e2fsprogs-1.47.3.tar.gz*
Somme de contrôle MD5 : `113d7a7ee0710d2a670a44692a35fd2e`

• Elfutils (0.193) — 11 695 Ko :
Page d'accueil : *https://sourceware.org/elfutils/*
Téléchargement : *https://sourceware.org/ftp/elfutils/0.193/elfutils-0.193.tar.bz2*
Somme de contrôle MD5 : `ceefa052ded950a4c523688799193a44`

• Expat (2.7.1) — 485 Ko :
Page d'accueil : *https://libexpat.github.io/*
Téléchargement : *https://github.com/libexpat/libexpat/releases/download/R_2_7_1/expat-2.7.1.tar.xz*
Somme de contrôle MD5 : `9f0c266ff4b9720beae0c6bd53ae4469`

• Expect (5.45.4) — 618 Ko :
Page d'accueil : *https://core.tcl.tk/expect/*
Téléchargement : *https://prdownloads.sourceforge.net/expect/expect5.45.4.tar.gz*
Somme de contrôle MD5 : `00fce8de158422f5ccd2666512329bd2`

• File (5.46) — 1 283 Ko :
Page d'accueil : *https://www.darwinsys.com/file/*
Téléchargement : *https://astron.com/pub/file/file-5.46.tar.gz*
Somme de contrôle MD5 : `459da2d4b534801e2e2861611d823864`

• Findutils (4.10.0) — 2 189 Ko :
Page d'accueil : *https://www.gnu.org/software/findutils/*
Téléchargement : *https://ftp.gnu.org/gnu/findutils/findutils-4.10.0.tar.xz*
Somme de contrôle MD5 : `870cfd71c07d37ebe56f9f4aaf4ad872`

• Flex (2.6.4) — 1 386 Ko :
Page d'accueil : *https://github.com/westes/flex*
Téléchargement : *https://github.com/westes/flex/releases/download/v2.6.4/flex-2.6.4.tar.gz*
Somme de contrôle MD5 : `2882e3179748cc9f9c23ec593d6adc8d`

• Flit-core (3.12.0) — 53 Ko :
Page d'accueil : *https://pypi.org/project/flit-core/*
Téléchargement : *https://pypi.org/packages/source/f/flit-core/flit_core-3.12.0.tar.gz*
Somme de contrôle MD5 : `c538415c1f27bd69cbbbf3cdd5135d39`

• Gawk (5.3.2) — 3 662 Ko :
Page d'accueil : *https://www.gnu.org/software/gawk/*
Téléchargement : *https://ftp.gnu.org/gnu/gawk/gawk-5.3.2.tar.xz*
Somme de contrôle MD5 : `b7014650c5f45e5d4837c31209dc0037`

• GCC (15.2.0) — 98 688 Ko :
Page d'accueil : *https://gcc.gnu.org/*
Téléchargement : *https://ftp.gnu.org/gnu/gcc/gcc-15.2.0/gcc-15.2.0.tar.xz*
Somme de contrôle MD5 : `b861b092bf1af683c46a8aa2e689a6fd`

• GDBM (1.26) — 1 198 Ko :
Page d'accueil : *https://www.gnu.org/software/gdbm/*
Téléchargement : *https://ftp.gnu.org/gnu/gdbm/gdbm-1.26.tar.gz*
Somme de contrôle MD5 : `aaa600665bc89e2febb3c7bd90679115`

• Gettext (0.26) — 9 926 Ko :
Page d'accueil : *https://www.gnu.org/software/gettext/*
Téléchargement : *https://ftp.gnu.org/gnu/gettext/gettext-0.26.tar.xz*
Somme de contrôle MD5 : `8e14e926f088e292f5f2bce95b81d10e`

• Glibc (2.42) — 19 464 Ko :
Page d'accueil : *https://www.gnu.org/software/libc/*
Téléchargement : *https://ftp.gnu.org/gnu/glibc/glibc-2.42.tar.xz*
Somme de contrôle MD5 : `23c6f5a27932b435cae94e087cb8b1f5`

Note

Les développeurs de Glibc maintiennent une *branche Git* qui contient des correctifs jugés utiles pour Glibc-2.42, mais malheureusement développés après la publication de 2.42. Les éditeurs de LFS publieront un avis de sécurité si une correction de sécurité est ajoutée à la branche, mais aucune action ne sera entreprise pour les autres correctifs ajoutés. Vous pouvez relire les correctifs vous-même et les incorporer si vous les jugez importants.

• GMP (6.3.0) — 2 046 Ko :
Page d'accueil : *https://www.gnu.org/software/gmp/*
Téléchargement : *https://ftp.gnu.org/gnu/gmp/gmp-6.3.0.tar.xz*
Somme de contrôle MD5 : `956dc04e864001a9c22429f761f2c283`

• Gperf (3.3) — 1 789 Ko :
Page d'accueil : *https://www.gnu.org/software/gperf/*
Téléchargement : *https://ftp.gnu.org/gnu/gperf/gperf-3.3.tar.gz*
Somme de contrôle MD5 : `31753b021ea78a21f154bf9eecb8b079`

• Grep (3.12) — 1 874 Ko :
Page d'accueil : *https://www.gnu.org/software/grep/*
Téléchargement : *https://ftp.gnu.org/gnu/grep/grep-3.12.tar.xz*
Somme de contrôle MD5 : `5d9301ed9d209c4a88c8d3a6fd08b9ac`

• Groff (1.23.0) — 7 259 Ko :
Page d'accueil : *https://www.gnu.org/software/groff/*
Téléchargement : *https://ftp.gnu.org/gnu/groff/groff-1.23.0.tar.gz*
Somme de contrôle MD5 : `5e4f40315a22bb8a158748e7d5094c7d`

• GRUB (2.12) — 6 524 Ko :
Page d'accueil : *https://www.gnu.org/software/grub/*
Téléchargement : *https://ftp.gnu.org/gnu/grub/grub-2.12.tar.xz*
Somme de contrôle MD5 : `60c564b1bdc39d8e43b3aab4bc0fb140`

• Gzip (1.14) — 865 Ko :
Page d'accueil : *https://www.gnu.org/software/gzip/*
Téléchargement : *https://ftp.gnu.org/gnu/gzip/gzip-1.14.tar.xz*
Somme de contrôle MD5 : `4bf5a10f287501ee8e8ebe00ef62b2c2`

• Iana-Etc (20250807) — 592 Ko :
Page d'accueil : *https://www.iana.org/protocols*
Téléchargement : *https://github.com/Mic92/iana-etc/releases/download/20250807/iana-etc-20250807.tar.gz*
Somme de contrôle MD5 : `de0a909103d4ff59d1424c5ec7ac9e4a`

• Inetutils (2.6) — 1 724 Ko :
Page d'accueil : *https://www.gnu.org/software/inetutils/*
Téléchargement : *https://ftp.gnu.org/gnu/inetutils/inetutils-2.6.tar.xz*
Somme de contrôle MD5 : `401d7d07682a193960bcdecafd03de94`

• Intltool (0.51.0) — 159 Ko :
Page d'accueil : *https://freedesktop.org/wiki/Software/intltool*
Téléchargement : *https://launchpad.net/intltool/trunk/0.51.0/+download/intltool-0.51.0.tar.gz*
Somme de contrôle MD5 : `12e517cac2b57a0121cda351570f1e63`

• IPRoute2 (6.16.0) — 910 Ko :
Page d'accueil : *https://www.kernel.org/pub/linux/utils/net/iproute2/*
Téléchargement : *https://www.kernel.org/pub/linux/utils/net/iproute2/iproute2-6.16.0.tar.xz*
Somme de contrôle MD5 : `80e1f91bf59d572acc15d5c6eb4f3e7c`

• Jinja2 (3.1.6) — 240 Ko :
Page d'accueil : *https://jinja.palletsprojects.com/en/3.1.x/*
Téléchargement : *https://pypi.org/packages/source/J/Jinja2/jinja2-3.1.6.tar.gz*
Somme de contrôle MD5 : `66d4c25ff43d1deaf9637ccda523dec8`

• Kbd (2.8.0) — 1 448 Ko :
Page d'accueil : *https://kbd-project.org/*
Téléchargement : *https://www.kernel.org/pub/linux/utils/kbd/kbd-2.8.0.tar.xz*
Somme de contrôle MD5 : `24b5d24f7483726b88f214dc6c77aa41`

• Kmod (34.2) — 434 Ko :
Page d'accueil : *https://github.com/kmod-project/kmod*
Téléchargement : *https://www.kernel.org/pub/linux/utils/kernel/kmod/kmod-34.2.tar.xz*
Somme de contrôle MD5 : `36f2cc483745e81ede3406fa55e1065a`

• Less (679) — 857 Ko :
Page d'accueil : *https://www.greenwoodsoftware.com/less/*
Téléchargement : *https://www.greenwoodsoftware.com/less/less-679.tar.gz*
Somme de contrôle MD5 : `0386dc14f6a081a94dfb4c2413864eed`

• LFS-Bootscripts (20250827) — 34 Ko :
Téléchargement : *https://www.linuxfromscratch.org/lfs/downloads/12.4/lfs-bootscripts-20250827.tar.xz*
Somme de contrôle MD5 : `3f661c64c2dfb55025767ed56074d059`

• Libcap (2.76) — 195 Ko :
Page d'accueil : *https://sites.google.com/site/fullycapable/*
Téléchargement : *https://www.kernel.org/pub/linux/libs/security/linux-privs/libcap2/libcap-2.76.tar.xz*
Somme de contrôle MD5 : `449ade7d620b5c4eeb15a632fbaa4f74`

• Libffi (3.5.2) — 1 390 Ko :
Page d'accueil : *https://sourceware.org/libffi/*
Téléchargement : *https://github.com/libffi/libffi/releases/download/v3.5.2/libffi-3.5.2.tar.gz*
Somme de contrôle MD5 : `92af9efad4ba398995abf44835c5d9e9`

• Libpipeline (1.5.8) — 1046 Ko :
Page d'accueil : *https://libpipeline.nongnu.org/*
Téléchargement : *https://download.savannah.gnu.org/releases/libpipeline/libpipeline-1.5.8.tar.gz*
Somme de contrôle MD5 : `17ac6969b2015386bcb5d278a08a40b5`

• Libtool (2.5.4) — 1 033 Ko :
Page d'accueil : *https://www.gnu.org/software/libtool/*
Téléchargement : *https://ftp.gnu.org/gnu/libtool/libtool-2.5.4.tar.xz*
Somme de contrôle MD5 : `22e0a29df8af5fdde276ea3a7d351d30`

• Libxcrypt (4.4.38) — 612 Ko :
Page d'accueil : *https://github.com/besser82/libxcrypt/*
Téléchargement : *https://github.com/besser82/libxcrypt/releases/download/v4.4.38/libxcrypt-4.4.38.tar.xz*
Somme de contrôle MD5 : `1796a5d20098e9dd9e3f576803c83000`

• Linux (6.16.1) — 149 042 Ko :
Page d'accueil : *https://www.kernel.org/*
Téléchargement : *https://www.kernel.org/pub/linux/kernel/v6.x/linux-6.16.1.tar.xz*
Somme de contrôle MD5 : `32d45755e4b39d06e9be58f6817445ee`

Note

Le noyau Linux est régulièrement mis à jour, souvent suite à la découverte de nouvelles failles de sécurité. Vous devriez utiliser la version la plus récente disponible du noyau, sauf si la page d'errata indique le contraire.

Pour les utilisateurs ayant un débit limité ou une bande passante chère, si vous souhaitez mettre à jour le noyau Linux, une version en ligne de commande du paquet et des correctifs peuvent être téléchargées séparément. Ceci peut économiser du temps ou de l'argent pour une mise à jour d'un niveau de correctif mineure (subsequent) à l'intérieur d'une version mineure.

• Lz4 (1.10.0) — 379 Ko :
Page d'accueil : *https://lz4.org/*
Téléchargement : *https://github.com/lz4/lz4/releases/download/v1.10.0/lz4-1.10.0.tar.gz*
Somme de contrôle MD5 : `dead9f5f1966d9ae56e1e32761e4e675`

• M4 (1.4.20) — 1 997 Ko :
Page d'accueil : *https://www.gnu.org/software/m4/*
Téléchargement : *https://ftp.gnu.org/gnu/m4/m4-1.4.20.tar.xz*
Somme de contrôle MD5 : `6eb2ebed5b24e74b6e890919331d2132`

• Make (4.4.1) — 2 300 Ko :
Page d'accueil : *https://www.gnu.org/software/make/*
Téléchargement : *https://ftp.gnu.org/gnu/make/make-4.4.1.tar.gz*
Somme de contrôle MD5 : `c8469a3713cbbe04d955d4ae4be23eeb`

• Man-DB (2.13.1) — 2 061 Ko :
Page d'accueil : *https://www.nongnu.org/man-db/*
Téléchargement : *https://download.savannah.gnu.org/releases/man-db/man-db-2.13.1.tar.xz*
Somme de contrôle MD5 : `b6335533cbeac3b24cd7be31fdee8c83`

• Man-pages (6.15) — 1 817 Ko :
Page d'accueil : *https://www.kernel.org/doc/man-pages/*
Téléchargement : *https://www.kernel.org/pub/linux/docs/man-pages/man-pages-6.15.tar.xz*
Somme de contrôle MD5 : `16f68d70139dd2bbcae4102be4705753`

• MarkupSafe (3.0.2) — 21 Ko :
Page d'accueil : *https://palletsprojects.com/p/markupsafe/*
Téléchargement : *https://pypi.org/packages/source/M/MarkupSafe/markupsafe-3.0.2.tar.gz*
Somme de contrôle MD5 : `cb0071711b573b155cc8f86e1de72167`

• Meson (1.8.3) — 2 282 :
Page d'accueil : *https://mesonbuild.com*
Téléchargement : *https://github.com/mesonbuild/meson/releases/download/1.8.3/meson-1.8.3.tar.gz*
Somme de contrôle MD5 : `08221d2f515e759686f666ff6409a903`

• MPC (1.3.1) — 756 Ko :
Page d'accueil : *https://www.multiprecision.org/*
Téléchargement : *https://ftp.gnu.org/gnu/mpc/mpc-1.3.1.tar.gz*
Somme de contrôle MD5 : `5c9bc658c9fd0f940e8e3e0f09530c62`

• MPFR (4.2.2) — 1 471 Ko :
Page d'accueil : *https://www.mpfr.org/*
Téléchargement : *https://ftp.gnu.org/gnu/mpfr/mpfr-4.2.2.tar.xz*
Somme de contrôle MD5 : `7c32c39b8b6e3ae85f25156228156061`

• Ncurses (6.5-20250809) — 3 703 Ko :
Page d'accueil : *https://www.gnu.org/software/ncurses/*
Téléchargement : *https://invisible-mirror.net/archives/ncurses/current/ncurses-6.5-20250809.tgz*
Somme de contrôle MD5 : `679987405412f970561cc85e1e6428a2`

• Ninja (1.13.1) — 286 Ko :
Page d'accueil : *https://ninja-build.org/*
Téléchargement : *https://github.com/ninja-build/ninja/archive/v1.13.1/ninja-1.13.1.tar.gz*
Somme de contrôle MD5 : `c35f8f55f4cf60f1a916068d8f45a0f8`

• OpenSSL (3.5.2) — 51 934 Ko :
Page d'accueil : *https://www.openssl-library.org/*
Téléchargement : *https://github.com/openssl/openssl/releases/download/openssl-3.5.2/openssl-3.5.2.tar.gz*
Somme de contrôle MD5 : `890fc59f86fc21b5e4d1c031a698dbde`

• Packaging (25.0) — 162 Ko :
Page d'accueil : *https://pypi.org/project/packaging/*
Téléchargement : *https://files.pythonhosted.org/packages/source/p/packaging/packaging-25.0.tar.gz*
Somme de contrôle MD5 : `ab0ef21ddebe09d1803575120d3f99f8`

• Patch (2.8) — 886 Ko :
Page d'accueil : *https://savannah.gnu.org/projects/patch/*
Téléchargement : *https://ftp.gnu.org/gnu/patch/patch-2.8.tar.xz*
Somme de contrôle MD5 : `149327a021d41c8f88d034eab41c039f`

• Perl (5.42.0) — 14 084 Ko :
Page d'accueil : *https://www.perl.org/*
Téléchargement : *https://www.cpan.org/src/5.0/perl-5.42.0.tar.xz*
Somme de contrôle MD5 : `7a6950a9f12d01eb96a9d2ed2f4e0072`

• Pkgconf (2.5.1) — 321 Ko :
Page d'accueil : *https://github.com/pkgconf/pkgconf*
Téléchargement : *https://distfiles.ariadne.space/pkgconf/pkgconf-2.5.1.tar.xz*
Somme de contrôle MD5 : `3291128c917fdb8fccd8c9e7784b643b`

• Procps (4.0.5) — 1 483 Ko :
Page d'accueil : *https://gitlab.com/procps-ng/procps/*
Téléchargement : *https://sourceforge.net/projects/procps-ng/files/Production/procps-ng-4.0.5.tar.xz*
Somme de contrôle MD5 : `90803e64f51f192f3325d25c3335d057`

• Psmisc (23.7) — 423 Ko :
Page d'accueil : *https://gitlab.com/psmisc/psmisc*
Téléchargement : *https://sourceforge.net/projects/psmisc/files/psmisc/psmisc-23.7.tar.xz*
Somme de contrôle MD5 : `53eae841735189a896d614cba440eb10`

• Python (3.13.7) — 22 236 Ko :
Page d'accueil : *https://www.python.org/*
Téléchargement : *https://www.python.org/ftp/python/3.13.7/Python-3.13.7.tar.xz*
Somme de contrôle MD5 : `256cdb3bbf45cdce7499e52ba6c36ea3`

• Documentation Python (3.13.7) — 10 183 Ko :
Téléchargement : *https://www.python.org/ftp/python/doc/3.13.7/python-3.13.7-docs-html.tar.bz2*
Somme de contrôle MD5 : `b84c0d81b2758398bb7f5b7411d3d908`

• Readline (8.3) — 3 340 Ko :
Page d'accueil : *https://tiswww.case.edu/php/chet/readline/rltop.html*
Téléchargement : *https://ftp.gnu.org/gnu/readline/readline-8.3.tar.gz*
Somme de contrôle MD5 : `25a73bfb2a3ad7146c5e9d4408d9f6cd`

• Sed (4.9) — 1 365 Ko :
Page d'accueil : *https://www.gnu.org/software/sed/*
Téléchargement : *https://ftp.gnu.org/gnu/sed/sed-4.9.tar.xz*
Somme de contrôle MD5 : `6aac9b2dbafcd5b7a67a8a9bcb8036c3`

• Setuptools (80.9.0) — 1 290 Ko :
Page d'accueil : *https://pypi.org/project/setuptools/*
Téléchargement : *https://pypi.org/packages/source/s/setuptools/setuptools-80.9.0.tar.gz*
Somme de contrôle MD5 : `82e1d67883b713f9493659b50d13b436`

• Shadow (4.18.0) — 2 293 Ko :
Page d'accueil : *https://github.com/shadow-maint/shadow/*
Téléchargement : *https://github.com/shadow-maint/shadow/releases/download/4.18.0/shadow-4.18.0.tar.xz*
Somme de contrôle MD5 : `30ef46f54363db1d624587be68794ef2`

• Sysklogd (2.7.2) — 474 Ko :
Page d'accueil : *https://www.infodrom.org/projects/sysklogd/*
Téléchargement : *https://github.com/troglobit/sysklogd/releases/download/v2.7.2/sysklogd-2.7.2.tar.gz*
Somme de contrôle MD5 : `af60786956a2dc84054fbf46652e515e`

• Systemd (257.8) — 16 002 Ko :
Page d'accueil : *https://www.freedesktop.org/wiki/Software/systemd/*
Téléchargement : *https://github.com/systemd/systemd/archive/v257.8/systemd-257.8.tar.gz*
Somme de contrôle MD5 : `25fe5d328e22641254761f1baa74cee0`

• Pages de manuel de Systemd (257.8) — 736 Ko :
Page d'accueil : *https://www.freedesktop.org/wiki/Software/systemd/*
Téléchargement : *https://anduin.linuxfromscratch.org/LFS/systemd-man-pages-257.8.tar.xz*
Somme de contrôle MD5 : `a44063e2ec0cf4adfd2ed5c9e9e095c5`

Note
L'équipe systemd de Linux From Scratch génère sa propre archive des pages de manuel à partir des sources de systemd. Cela est fait pour éviter des dépendances inutiles.

• SysVinit (3.14) — 236 Ko :
Page d'accueil : *https://savannah.nongnu.org/projects/sysvinit*
Téléchargement : *https://github.com/slicer69/sysvinit/releases/download/3.14/sysvinit-3.14.tar.xz*
Somme de contrôle MD5 : `bc6890b975d19dc9db42d0c7364dd092`

• Tar (1.35) — 2 263 Ko :
Page d'accueil : *https://www.gnu.org/software/tar/*
Téléchargement : *https://ftp.gnu.org/gnu/tar/tar-1.35.tar.xz*
Somme de contrôle MD5 : `a2d8042658cfd8ea939e6d911eaf4152`

• Tcl (8.6.16) — 11 406 Ko :
Page d'accueil : *https://tcl.sourceforge.net/*
Téléchargement : *https://downloads.sourceforge.net/tcl/tcl8.6.16-src.tar.gz*
Somme de contrôle MD5 : `eaef5d0a27239fb840f04af8ec608242`

• Documentation de Tcl (8.6.16) — 1 169 Ko :
Téléchargement : *https://downloads.sourceforge.net/tcl/tcl8.6.16-html.tar.gz*
Somme de contrôle MD5 : `750c221bcb6f8737a6791c1fbe98b684`

• Texinfo (7.2) — 6 259 Ko :
Page d'accueil : *https://www.gnu.org/software/texinfo/*
Téléchargement : *https://ftp.gnu.org/gnu/texinfo/texinfo-7.2.tar.xz*
Somme de contrôle MD5 : `11939a7624572814912a18e76c8d8972`

• Time Zone Data (2025b) — 454 Ko :
Page d'accueil : *https://www.iana.org/time-zones*
Téléchargement : *https://www.iana.org/time-zones/repository/releases/tzdata2025b.tar.gz*
Somme de contrôle MD5 : `ad65154c48c74a9b311fe84778c5434f`

• Udev-lfs Archive Tar (udev-lfs-20230818) — 10 Ko :
Téléchargement : *https://anduin.linuxfromscratch.org/LFS/udev-lfs-20230818.tar.xz*
Somme de contrôle MD5 : `acd4360d8a5c3ef320b9db88d275dae6`

• Util-linux (2.41.1) — 9 382 Ko :
Page d'accueil : *https://git.kernel.org/pub/scm/utils/util-linux/util-linux.git/*
Téléchargement : *https://www.kernel.org/pub/linux/utils/util-linux/v2.41/util-linux-2.41.1.tar.xz*
Somme de contrôle MD5 : `7e5e68845e2f347cf96f5448165f1764`

• Vim (9.1.1629) — 18 317 Ko :
Page d'accueil : *https://www.vim.org*
Téléchargement : *https://github.com/vim/vim/archive/v9.1.1629/vim-9.1.1629.tar.gz*
Somme de contrôle MD5 : `4f856c3233c1c4570bc17572e4f9e8e4`

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

• Wheel (0.46.1) — 54 Ko :
Page d'accueil : *https://pypi.org/project/wheel/*
Téléchargement : *https://pypi.org/packages/source/w/wheel/wheel-0.46.1.tar.gz*
Somme de contrôle MD5 : `65e09ee84af36821e3b1e9564aa91bd5`

• XML::Parser (2.47) — 276 Ko :
Page d'accueil : *https://github.com/chorny/XML-Parser*
Téléchargement : *https://cpan.metacpan.org/authors/id/T/TO/TODDR/XML-Parser-2.47.tar.gz*
Somme de contrôle MD5 : `89a8e82cfd2ad948b349c0a69c494463`

• Xz Utils (5.8.1) — 1 428 Ko :
Page d'accueil : *https://tukaani.org/xz*
Téléchargement : *https://github.com//tukaani-project/xz/releases/download/v5.8.1/xz-5.8.1.tar.xz*
Somme de contrôle MD5 : `cf5e1feb023d22c6bdaa30e84ef3abe3`

• Zlib (1.3.1) — 1 478 Ko :
Page d'accueil : *https://zlib.net/*
Téléchargement : *https://zlib.net/fossils/zlib-1.3.1.tar.gz*
Somme de contrôle MD5 : `9855b6d802d7fe5b7bd5b196a2271655`

• Zstd (1.5.7) — 2 378 Ko :
Page d'accueil : *https://facebook.github.io/zstd/*
Téléchargement : *https://github.com/facebook/zstd/releases/download/v1.5.7/zstd-1.5.7.tar.gz*
Somme de contrôle MD5 : `780fc1896922b1bc52a4e90980cdda48`

Taille totale de ces paquets : environ NaN Mo

3.3. Correctifs requis

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

• Bzip2 Correctif documentation — 1.6 Ko :
Téléchargement : *https://www.linuxfromscratch.org/patches/lfs/12.4/bzip2-1.0.8-install_docs-1.patch*
Somme de contrôle MD5 : `6a5ac7e89b791aae556de0f745916f7f`

• Corrections en amont de Coreutils — 4.1 Ko :
Téléchargement : *https://www.linuxfromscratch.org/patches/lfs/12.4/coreutils-9.7-upstream_fix-1.patch*
Somme de contrôle MD5 : `96382a5aa85d6651a74f94ffb61785d9`

• Coreutils Correctif pour l'internationalisation — 159 Ko:
Téléchargement : *https://www.linuxfromscratch.org/patches/lfs/12.4/coreutils-9.7-i18n-1.patch*
Somme de contrôle MD5 : `33ebfad32b2dfb8417c3335c08671206`

• Correctif Expect pour GCC15 — 12 Ko :
Téléchargement : *https://www.linuxfromscratch.org/patches/lfs/12.4/expect-5.45.4-gcc15-1.patch*
Somme de contrôle MD5 : `0ca4d6bb8d572fbcdb13cb36cd34833e`

• Glibc correctif FHS — 2.8 Ko :
Téléchargement : *https://www.linuxfromscratch.org/patches/lfs/12.4/glibc-2.42-fhs-1.patch*
Somme de contrôle MD5 : `9a5997c3452909b1769918c759eff8a2`

• Correctif réparant Kdb Backspace/Delete — 12 Ko :
Téléchargement : *https://www.linuxfromscratch.org/patches/lfs/12.4/kbd-2.8.0-backspace-1.patch*
Somme de contrôle MD5 : `f75cca16a38da6caa7d52151f7136895`

• Correctif consolidé de SysVinit — 2.5 Ko :
Téléchargement : *https://www.linuxfromscratch.org/patches/lfs/12.4/sysvinit-3.14-consolidated-1.patch*
Somme de contrôle MD5 : `3af8fd8e13cad481eeeaa48be4247445`

Taille totale de ces correctifs : environ 194 Ko

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

Chapitre 4. Dernières préparations

4.1. Introduction

Dans ce chapitre, nous allons effectuer quelques tâches supplémentaires pour préparer la construction du système temporaire. Nous allons créer un ensemble de répertoires dans `$LFS` (pour l'installation des outils temporaires), ajouter un utilisateur non privilégié, et créer un environnement de construction adéquat pour cet utilisateur. Nous allons également expliquer l'unité de temps (« SBU ») utilisée pour mesurer la durée de construction des paquets LFS et donner quelques informations sur les suites de tests des paquets.

4.2. Créer une structure des répertoires limitée dans le système de fichiers LFS

Dans cette section, nous alimenteront le système de fichiers LFS de tous les éléments qui composeront le système Linux final. La première étape consiste à créer une arborescence de répertoires limitée pour que les programmes compilés dans le Chapitre 6 (ainsi que glibc et libstdc++ dans le Chapitre 5) puissent être installés à leur emplacement final. De cette manière, les programmes temporaires seront remplacés lorsque les versions finales seront reconstruites dans le Chapitre 8.

Pour créer la structure des répertoires requise, exécutez les commandes suivantes en tant qu'utilisateur `root` :

```
mkdir -pv $LFS/{etc,var} $LFS/usr/{bin,lib,sbin}

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

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

Les programmes de Chapitre 6 seront compilés avec un compilateur croisé (plus d'informations dans la section Remarques techniques sur la chaîne de compilation), qui sera installé dans un répertoire spécifique séparé des autres programmes. Pour créer ce répertoire, exécutez la commande :

```
mkdir -pv $LFS/tools
```

Note

Les éditeurs de LFS ont choisi délibérément de ne pas utiliser de répertoire `/usr/lib64`. Plusieurs mesures sont prises pour s'assurer que la chaine d'outils ne l'utilisera pas. Si pour une quelconque raison ce répertoire apparaît (soit parce que vous avez fait une erreur en suivant les instructions, soit parce que vous avez installé un paquet binaire qui l'a créé après avoir fini LFS), cela pourrait casser votre système. Vous devriez toujours vous assurer que ce répertoire n'existe pas.

4.3. Ajouter l'utilisateur LFS

Lorsque vous êtes connecté en tant qu'utilisateur `root`, une simple erreur peut endommager, voire détruire, votre système. Les paquets des deux prochains chapitres sont donc construits en tant qu'utilisateur non privilégié. Vous pouvez bien sûr utiliser votre propre nom d'utilisateur, mais pour faciliter la configuration d'un environnement de travail propre, créez un nouvel utilisateur `lfs` comme membre d'un nouveau groupe (aussi nommé `lfs`) et utilisez cet utilisateur pour exécuter les commandes 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`
Fait de **bash** le shell par défaut de l'utilisateur `lfs`.

`-g lfs`
Ajoute l'utilisateur `lfs` au groupe `lfs`.

`-m`
Crée un répertoire personnel pour `lfs`.

`-k /dev/null`
Empêche toute copie possible de fichiers provenant du répertoire squelette (par défaut, `/etc/skel`) en modifiant son emplacement par le périphérique spécial null.

`lfs`
C'est le nom du nouvel utilisateur.

Si vous souhaitez vous connecter en tant que `lfs` ou passer d'un utilisateur non-`root` à `lfs` (et non pas de l'utilisateur `root` à l'utilisateur `lfs`, ce qui ne nécessite pas de mot de passe pour `lfs`), vous devez définir le mot de passe de `lfs`. Exécutez la commande suivante en tant qu'utilisateur `root` pour configurer un mot de passe :

```
passwd lfs
```

Donnez à `lfs` un accès complet aux répertoires de `$LFS` en indiquant que `lfs` est le propriétaire du répertoire :

```
chown -v lfs $LFS/{usr{,/*},var,etc,tools}
case $(uname -m) in
  x86_64) chown -v lfs $LFS/lib64;;
esac
```

Note

Sur certains systèmes hôtes, la commande **su** suivante ne s'exécute pas correctement et place la connexion vers l'utilisateur `lfs` en tâche de fond. Si l'invite de commande « lfs:~$ » n'apparaît pas immédiatement, exécutez la commande **fg** pour corriger le problème.

Ensuite, connectez-vous en tant que `lfs`. Vous pouvez le faire en vous connectant en tant qu'utilisateur `lfs`, soit via une console virtuelle, soit avec la commande de substitution d'utilisateur suivante :

```
su - lfs
```

Le « - » indique à **su** qu'il doit lancer un shell de connexion. Vous trouverez la différence entre un shell de connexion et un shell sans connexion dans la page de manuel *bash(1)* et **info bash**.

4.4. Configurer l'environnement

Configurez un bon environnement de travail en créant deux nouveaux fichiers de démarrage pour le shell **bash**. En tant qu'utilisateur `lfs`, exécutez 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 qu'utilisateur `lfs` ou transféré vers l'utilisateur `lfs` en utilisant une commande **su** avec l'option « - », le shell initial est un shell de *connexion* qui lit le fichier `/etc/profile` de l'hôte (contenant probablement quelques configurations et variables d'environnement) puis `.bash_profile`. La commande **exec env -i.../bin/bash** dans le fichier `.bash_profile` remplace le shell en cours par un nouveau shell ayant un environnement complètement vide en dehors des variables `HOME`, `TERM`, et `PS1`. Cela garantit qu'aucune variable d'environnement non souhaitée et potentiellement dangereuse provenant du système hôte ne fuit dans l'environnement de construction. La technique utilisée ici permet d'obtenir un environnement propre.

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

```
cat > ~/.bashrc << "EOF"
set +h
umask 022
LFS=/mnt/lfs
LC_ALL=POSIX
LFS_TGT=$(uname -m)-lfs-linux-gnu
PATH=/usr/bin
if [ ! -L /bin ]; then PATH=/bin:$PATH; fi
PATH=$LFS/tools/bin:$PATH
CONFIG_SITE=$LFS/usr/share/config.site
export LFS LC_ALL LFS_TGT PATH CONFIG_SITE
EOF
```

Voici la signification des paramètres du fichier `.bashrc`

`set +h`

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

`umask 022`

Cela configure l'umask comme expliqué dans Section 2.6, « Définition de la variable $LFS et du Umask. »

`LFS=/mnt/lfs`

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

`LC_ALL=POSIX`

La variable `LC_ALL` contrôle les paramètres linguistiques de certains programmes pour que leurs messages suivent les conventions du pays spécifié. Définir `LC_ALL` à « POSIX » ou « C » (les deux étant équivalents) garantit que tout fonctionnera comme prévu dans l'environnement de compilation croisée.

`LFS_TGT=$(uname -m)-lfs-linux-gnu`

La variable `LFS_TGT` initialise une description de machine personnalisée mais compatible lors de la construction de notre compilateur croisé, de notre éditeur de liens et lors de la compilation croisée de notre chaîne de compilation temporaire. Vous trouverez plus d'informations dans la Remarques techniques sur la chaîne de compilation.

`PATH=/usr/bin`

De nombreuses distributions récentes de Linux fusionnent `/bin` et `/usr/bin`. Lorsque c'est le cas, il vaut mieux paramétrer la variable `PATH` standard sur `/usr/bin` pour l'environnement Chapitre 6. Lorsque ce n'est pas le cas, la ligne suivante ajoute `/bin` au chemin de recherche.

`if [ ! -L /bin ]; then PATH=/bin:$PATH; fi`

Si `/bin` n'est pas un lien symbolique, il doit être ajouté à la variable `PATH`.

`PATH=$LFS/tools/bin:$PATH`

En plaçant `$LFS/tools/bin` au début du `PATH` standard, le compilateur croisé installé au début du Chapitre 5 est repéré par le shell immédiatement après son installation. Ceci, combiné à la désactivation du hachage, limite le risque que le compilateur de l'hôte ne soit utilisé à la place du compilateur croisé.

`CONFIG_SITE=$LFS/usr/share/config.site`

Dans le Chapitre 5 et le Chapitre 6, si cette variable n'est pas initialisée, les scripts **configure** peuvent essayer de charger des bouts de configuration de certaines distributions dans `/usr/share/config.site` sur le système hôte. Changer ce chemin permet d'éviter une contamination potentielle par l'hôte.

export ...

Bien que les commandes précédentes aient configuré certaines variables, pour les rendre visibles dans les sous-shells nous les exportons.

Important

Plusieurs distributions commerciales ajoutent une instance non documentée de `/etc/bash.bashrc` à l'initialisation de **bash**. Ce fichier peut modifier l'environnement de l'utilisateur `lfs` d'une manière qui peut affecter la construction de paquets critiques de LFS. Pour vous assurer que l'environnement de l'utilisateur `lfs` est propre, vérifiez la présence de `/etc/bash.bashrc` et, s'il est présent, déplacez-le ailleurs. En tant qu'utilisateur `root`, exécutez la commande :

```
[ ! -e /etc/bash.bashrc ] || mv -v /etc/bash.bashrc /etc/bash.bashrc.NOUSE
```

Lorsque l'utilisateur `lfs` n'est plus nécessaire (au début du Chapitre 7), vous pouvez restaurer `/etc/bash.bashrc` (si vous le souhaitez).

Notez que le paquet Bash de LFS que nous construisons dans la Section 8.36, « Bash-5.3 » n'est pas configuré pour charger ou exécuter `/etc/bash.bashrc`, rendant ce fichier sur un système LFS complet.

Pour de nombreux systèmes modernes avec plusieurs processeurs (ou cœurs) le temps de compilation pour un paquet peut être réduit en effectuant un « make parallèle » en disant au programme make combien de processeurs sont disponibles via une option de la ligne de commande ou une variable d'environnement. Par exemple, un processeur Intel Core i9-13900K a 8 cœurs P (performance) et 16 cœurs E (efficacité) et un cœur P peut faire tourner simultanément deux fils d'exécution donc chaque cœur P est modélisé par deux cœurs logiques pour la noyau Linux. Il y a donc 32 cœurs logiques au total. Une manière simple d'utiliser tous ces cœurs logiques est de permettre à **make** de démarrer 32 tâches de construction. Vous pouvez le faire en passant l'option *-j32* à **make** :

```
make -j32
```

Ou en indiquant la variable d'environnement `MAKEFLAGS` dont le contenu sera automatiquement utilisé par **make** comme s'il s'agissait d'options de la ligne de commande :

```
export MAKEFLAGS=-j32
```

Important

Ne passez jamais l'option *-j* sans un nombre à **make** et n'indiquez pas cette option dans `MAKEFLAGS`. Cela premettra à **make** de démarrer un nombre de tâche infini et causera des problèmes de stabilité sur le système.

Pour utiliser tous les cœurs logiques disponibles pour construire les paquets dans Chapitre 5 et Chapitre 6, configurez maintenant `MAKEFLAGS` dans `.bashrc` :

```
cat >> ~/.bashrc << "EOF"
export MAKEFLAGS=-j$(nproc)
EOF
```

Remplacez *$(nproc)* par le nombre de cœurs logiques que vous voulez utiliser si vous ne voulez pas utiliser tous les cœurs logiques.

Enfin, pour avoir un environnement complètement préparé pour la construction des outils temporaires, forcez le shell **bash** à lire le profil de l'utilisateur tout juste créé :

```
source ~/.bash_profile
```

4.5. À propos des SBU

Beaucoup de personnes souhaitent savoir à l'avance combien de temps la compilation et l'installation de chaque paquet va prendre. Linux From Scratch est construit sur tellement de systèmes différents qu'il est impossible d'en estimer la durée précise. Le plus gros paquet (gcc) met approximativement 5 minutes sur les systèmes les plus rapides mais peut prendre plusieurs jours sur les moins rapides ! Plutôt que de donner des délais précis, nous utilisons les SBU (Standard Build Units).

La mesure des SBU fonctionne de cette manière : le premier paquet que vous compilez dans ce livre est binutils dans le Chapitre 5. Le délai de compilation de ce paquet avec un unique cœur correspond à ce que nous appelons un SBU. Tous les autres délais de compilation sont exprimés par rapport à celui-ci.

Par exemple, prenez un paquet spécifique dont le temps de compilation correspond à 4,5 SBU. Cela signifie que s'il vous a fallu 4 minutes pour compiler et installer la première passe de Binutils, alors vous savez qu'il faudra *environ* 18 minutes pour construire ce paquet. Heureusement, la plupart des délais de construction sont bien plus courts que celui de binutils.

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

Sur certains systèmes récents, la carte mère peut contrôler la vitesse de l'horloge système. Cela peut être contrôlé avec une commande comme **powerprofilesctl**. Elle n'est pas disponible dans LFS, mais peut être disponible sur la distribution hôte. Après la complétion de LFS, elle peut être ajoutée au système avec les procédures de la page *BLFS sur power-profiles-daemon*. Avant de mesurer le temps de construction d'un paquet il est recommandé d'utiliser un profil d'énergie système configuré pour les performances maximum (et la consommation d'énergie maximum). Sinon les SBU mesurés peuvent être imprécis car le système peut réagir différemment lors de la construction de binutils-pass1 ou d'autres paquets. Soyez conscient qu'un grand nombre d'imprécisions peut toujours apparaître même si le même profil est utilisé pour les deux paquets car le système peut répondre plus lentement si le système est en attente lorsque la procédure de construction démarre. Utiliser le profil d'énergie « performance » minimisera ce problème. Évidemment, cela accélérera également la construction de LFS.

Si **powerprofilesctl** est disponible, exécutez la commande **powerprofilesctl set performance** pour choisir le profil `performance`. Certaines distributions fournissent la commande **tuned-adm** pour gérer les profils au lieu de **powerprofilesctl**. Sur ces distributions, exécutez la commande **tuned-adm profile throughput-performance** pour choisir le profil `throughput-performance`.

Note

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

Les délais présentés ici pour tous les paquets (sauf binutils-pass1 qui se base sur un seul cœur) se basent sur l'utilisation de quatre cœurs (-j4). Les délais du chapitre 8 comprennent aussi le temps nécessaire à lancer les tests de non-régression pour les paquets, sauf mention contraire.

4.6. À propos des suites de tests

La plupart des paquets disposent d'une suite de tests. Lancer cette suite de tests pour un paquet tout juste construit est généralement une bonne idée car elle permet de faire un « sanity check », qui indique si tout a été compilé correctement. Une suite de tests réussissant l'ensemble des vérifications prouve généralement que le paquet fonctionne comme le développeur le souhaitait. Néanmoins, elle ne garantit pas que le paquet ne contient pas de bugs.

Certaines suites de tests sont plus importantes que d'autres. Par exemple, les suites de tests des paquets de chaîne de compilation — GCC, binutils, et glibc — sont de la plus haute importance étant donné leur rôle central dans un système fonctionnel. Les suites de tests pour GCC et glibc peuvent mettre beaucoup de temps à se terminer, surtout sur une machine lente, mais elles sont fortement recommandées.

Note

Lancer les suites de tests dans le Chapitre 5 et le Chapitre 6 est inutile, car ces programmes sont compilés avec un compilateur croisé, donc ils ne sont pas sensés être exécutables sur l'hôte de construction.

Un problème récurrent lors du lancement des suites de test pour binutils et GCC est de manquer de pseudo-terminaux (PTY). Cela peut entraîner un nombre élevé d'échecs dans les tests, et ce pour plusieurs raisons, mais la plus probable est que le système hôte ne dispose pas d'un système de fichiers `devpts` configuré correctement. Pour plus de détails, voir *https://fr.linuxfromscratch.org/faq/lfs#no-ptys*.

Parfois, les suites de test des paquets échouent pour diverses raisons, que les développeurs ont identifié et considéré comme non critiques. Consultez les logs sur *https://www.linuxfromscratch.org/lfs/build-logs/12.4/* pour vérifier si ces échecs sont susceptibles d’avoir lieu. Ce site est valable pour tous les tests effectués dans ce livre.

Partie III. Construction des outils croisés LFS et des outils temporaires

Informations préliminaires importantes

Introduction

Cette partie est divisée en trois étapes : la première construit un compilateur croisé et ses bibliothèques associées ; la seconde utilise cette chaîne d'outils croisée pour construire plusieurs outils de sorte à ce qu'ils soient isolés de la distribution hôte ; la troisième entre dans l'environnement chroot, qui améliore encore plus l'isolation et construit le reste des outils requis pour construire le système final.

Important

C'est dans cette partie qu'un vrai travail de construction d'un nouveau système peut commencer. Il est recommande d'être attentif et de suivre avec précision les instructions du livre. Mettez de côté toute envie de finir votre construction rapidement. Vous devez essayer de comprendre ce que les commandes font et éviter de les taper à l'aveugle. Lisez la documentation quand vous ne comprenez pas quelque chose. Gardez aussi une trace de ce que vous tapez et des sorties des commandes, en les envoyant dans un fichier, avec l'outil **tee**, qui permet de mieux diagnostiquer les problèmes qui pourraient survenir.

La prochaine section est une introduction technique au processus de construction, tandis que la suivante contient des instructions générales **très importantes**.

Remarques techniques sur la chaîne de compilation

Cette section explique certains détails logiques et techniques de la méthode de construction. Il n'est pas essentiel de tout comprendre immédiatement. La plupart de ces informations seront plus claires une fois votre première construction complète terminée. Cette section peut servir de référence à tout moment lors du processus de construction.

Le but global du Chapitre 5 et du Chapitre 6 est de générer un espace temporaire comportant un ensemble d'outils connus et approuvés, qui sont isolés du système hôte. En utilisant la commande **chroot**, les compilations réalisées dans le reste des chapitres se cantonneront à cet environnement, en assurant une construction du système LFS cible propre et sans faille. Le processus de construction a été conçu pour minimiser les risques auprès des nouveaux lecteurs et pour fournir une valeur éducative maximale.

Le processus de construction se base sur la *compilation croisée*. La compilation croisée est normalement utilisée pour construire un compilateur ainsi que sa chaîne de compilation sur une machine différente de celle utilisée pour la construction. Ce n'est pas nécessaire pour LFS, puisque la machine sur laquelle le nouveau système est construit est la même que celle utilisée pour la construction. Mais le grand avantage de la compilation croisée, c'est que tout ce qui est compilé est indépendant de l'environnement hôte.

À propos de la compilation croisée

Note

Le livre LFS n'est pas (et ne contient pas) un tutoriel générique sur la construction d'une chaîne de compilation croisée (ou native). N'utilisez pas les commandes de ce livre pour construire une chaîne de compilation croisée autre que celle utilisée pour LFS, à moins de vraiment comprendre ce que vous faites.

Nous savons que GCC passe 2 cassera la chaîne d'outils croisée. Nous ne considérons pas cela comme un bogue car GCC passe 2 est le dernier paquet à être compilé de manière croisée dans le livre, et nous ne le « corrigerons » pas tant que nous n'aurons pas vraiment besoin de compiler certains paquets de manière croisée après GCC passe 2.

La compilation croisée utilise certains concepts qui mériteraient une section à part. Même si vous pouvez passer cette section lors de votre première lecture, nous vous recommandons fortement d'y revenir plus tard pour bien comprendre le processus de construction.

Définissons d'abord certains termes utilisés dans ce contexte.

La construction (build)
: est la machine où nous construisons les programmes. Cette machine est appelée « hôte » dans les autres sections.

L’hôte (host)
: est la machine ou le système où les programmes seront lancés. Le terme « hôte » n’a pas la même définition dans les autres sections.

La cible (target)
: est seulement utilisée pour les compilateurs. C'est la machine pour laquelle le compilateur produit du code. Elle peut être différente de l’hôte et de la construction.

Par exemple, imaginons le scénario suivant (parfois appelé « Canadian Cross ») : nous avons un compilateur sur une machine lente, que nous appellerons A, et le compilateur ccA. Nous avons également une machine rapide (B), mais aucun compilateur pour (B). Nous voulons produire du code pour une troisième machine, lente cette fois (C). Il y a trois étapes à suivre pour construire un compilateur destiné à la machine C.

Étape	Construction	Hôte	Cible	Action
1	A	A	B	Construire un compilateur croisé cc1 avec ccA sur la machine A.
2	A	B	C	Construire un compilateur croisé cc2 avec cc1 sur la machine A.
3	B	C	C	Construire le compilateur ccC avec cc2 sur la machine B.

Ensuite, tous les programmes requis par la machine C peuvent être compilés avec cc2 sur la machine rapide B. À moins que B ne puisse lancer les programmes produits pour C, il n'existe aucun moyen de tester les nouveaux programmes construits avant de les lancer sur la machine C. Par exemple, pour tester ccC, nous pouvons ajouter une quatrième étape :

Étape	Construction	Hôte	Cible	Action
4	C	C	C	Reconstruire et tester ccC avec lui-même sur la machine C.

Dans l'exemple ci-dessus, seuls cc1 et cc2 sont des compilateurs croisés, c'est-à-dire qu'ils produisent du code pour une machine différente de celle sur laquelle ils tournent. Les autres compilateurs ccA et ccC produisent du code pour la machine sur laquelle ils tournent. Ces compilateurs sont appelés compilateurs *natifs*.

Implémentation de la compilation croisée dans LFS

Note

Tous les paquets compilés de manière croisée dans ce livre utilisent un système de construction basé sur autoconf. Le système de construction basé sur autoconf accepte des types de systèmes de la forme cpu-fabriquant-noyau-os, nommés triplets systèmes. Comme le champ fabriquant est souvent inutile, autoconf vous autorise à ne pas le renseigner.

Le lecteur attentif se demandera pourquoi un « triplet » désigne un nom à quatre composantes. Le champ du noyau et de l'os ont d'abord commencé comme un seul champ « système ». Cette forme à trois champs est toujours valide de nos jours pour certains systèmes, par exemple `x86_64-unknown-freebsd`. Mais deux systèmes peuvent partager le même noyau et être trop différents pour utiliser le même triplet pour les désigner. Par exemple Android qui tourne sur les téléphones est complètement différent d'Ubuntu sur un serveur ARM64, même s'ils utilisent tous les deux le même type de CPU (ARM64) et utilisent le même noyau (Linux).

Sans une couche d'émulation, vous ne pouvez pas lancer un exécutable pour un serveur sur un téléphone portable et vice-versa. Donc le champ « système » a été divisé en les champs noyau et os, pour désigner trois systèmes sans ambiguïté. Dans notre exemple, le système Android est appelé `aarch64-unknown-linux-android` et le système Ubuntu est appelé `aarch64-unknown-linux-gnu`.

Le mot « triplet » reste ancré dans le vocabulaire informatique. Pour déterminer simplement votre triplet système, vous pouvez lancer le script **config.guess** fournit avec les sources de nombreux paquets. Désarchivez les sources de binutils, lancez le script **`./config.guess`**, et notez la sortie. Par exemple pour un processeur Intel 32 bits la sortie sera *i686-pc-linux-gnu*. Sur un système 64 bits elle sera *x86_64-pc-linux-gnu*. Sur la plupart des systèmes Linux la commande encore plus simple **gcc -dumpmachine** vous donner cette information.

Faites également attention au nom de l'éditeur de liens dynamiques de la plateforme, souvent appelé chargeur dynamique (à ne pas confondre avec l'éditeur de liens standard **ld** qui fait partie de binutils). Le chargeur dynamique fourni par glibc trouve et charge les bibliothèques partagées nécessaires à l'exécution d'un programme, prépare le programme, puis l'exécute.Le nom du chargeur dynamique pour une machine Intel 32 bits sera `ld-linux.so.2` (`ld-linux-x86-64.so.2` pour les systèmes 64 bits). Pour déterminer le nom du chargeur dynamique, inspectez un binaire au hasard sur le système hôte en exécutant : **`readelf -l <nom du binaire> | grep interpreter`** et récupérez le résultat. La référence officielle qui couvre toutes les plateformes se trouve sur *une page wiki de Glibc*.

Il y a deux points clés pour une compilation croisée :

- Lorsqu'on produit et traite du code machine censé être exécuté sur « l'hôte », la chaîne d'outils croisée doit être utilisée. Remarquez que la chaîne d'outils native de « la construction » peut quand même être invoquée pour générer du code machine censée être exécuté sur « la construction ». Par exemple, le système de construction peut compiler un générateur avec la chaîne d'outils native, puis générer un fichier source C avec le générateur, puis enfin compiler le fichier source C avec la chaîne d'outils croisée pour que le code généré puisse être exécuté sur « l'hôte ».

  Avec un système de construction basé sur autoconf, ce prérequis est assuré en utilisant le paramètre *--host* pour spécifier le triplet « hôte ». Avec ce paramètre, le système de construction utilisera les composants de la chaîne d'outils préfixés de `<the host triplet>` pour générer et traiter le code machine pour « l'hôte » ; p. ex. le compilateur sera ***<the host triplet>*-gcc** et l'outil **readelf** sera ***<the host triplet>*-readelf**.

- Le système de construction ne devrait pas essayer d'exécuter du code machine généré censé être exécuté sur « l'hôte ». Par exemple, lorsque vous construisez un utilitaire nativement, sa page de manuel sera générée en exécutant l'utilitaire avec le paramètre *--help* et en traitant la sortie, mais en général ce n'est pas possible pour une compilation croisée puisque l'utilitaire ne se lancera pas sur « la construction » : il est évidemment impossible d'exécuter du code machine ARM64 sur un CPU x86 (sans un émulateur).

  Avec un système de construction basé sur autoconf, ce prérequis est satisfait dans « le mode de compilation croisée » où les fonctionnalités facultatives qui nécessitent d'exécuter du code machine pour « l'hôte » pendant la construction sont désactivées. Lorsque le triplet « hôte » est spécifié explicitement, « le mode de compilation croisée » est activée si et seulement si le script **configure** échoue à exécuter un simple programme compilé vers le code machine de « l'hôte », ou si le triplet de « la construction » est explicitement spécifié par le paramètre *--build* et qu'il est différent du triple de « l'hôte ».

Pour compiler un paquet de manière croisée pour le système LFS temporaire, le nom du triplet système est légèrement ajusté en modifiant le champ "fabriquant" dans la variable `LFS_TGT` pour qu'il indique "lfs" et `LFS_TGT` est ensuite spécifié comme triplet « hôte » via *--host*, pour que la chaîne d'outils croisée puisse être utilisée pour générer et traiter le code machine qui s'exécute dans le système LFS temporaire. De plus, nous devons activer le « mode de compilation croisée » : bien que le code machine de « l'hôte », c.-à-d. le code machine pour le système LFS temporaire, puisse s'exécuter sur le CPU actuel, il peut référencer des bibliothèques qui ne sont pas disponibles sur « la construction » (la distribution hôte), ou bien du code ou de la donnée pourrait ne pas exister ou être défini différemment dans ces bibliothèques même si elles existent. Lors de la compilation croisée d'un paquet pour le système LFS temporaire, nous ne pouvons pas nous reposer sur le script **configure** pour détecter ce problème avec le programme simple : il utilise seulement quelques composants de la `libc` que la `libc` de la distribution hôte fournit probablement (à moins que la distribution hôte n'utilise une implémentation de la `libc` différente, comme Musl). Ce programme n'échouera donc pas, contrairement aux programmes vraiment utiles. Ainsi, nous devons spécifier explicitement le triplet de « la construction » pour activer le « mode de compilation croisée ». La valeur que nous utilisons est simplement la valeur par défaut, c.-à-d. le triplet système original de la sortie de **config.guess**, mais le « mode de compilation croisée » dépend d'une définition explicite comme nous l'avons indiqué.

Nous utilisons l'option *--with-sysroot* lors de la construction de l'éditeur des liens croisé et du compilateur croisé, pour leur dire où trouver les fichiers requis pour « l'hôte ». Cela s'assure presque qu'aucun autre programme construit dans Chapitre 6 puisse se lier aux bibliothèques de « la construction ». Nous utilisons le mot « presque » car **libtool**, une enveloppe de « compatibilité » du compilateur et de l'éditeur des liens pour les systèmes de construction basés sur autoconf, peut essayer d'être trop intelligent et passer par erreur des options qui permettent à l'éditeur des liens de trouver les bibliothèques de « la construction ». Pour éviter ce désastre, nous devons supprimer les fichiers d'archives libtool (`.la`) et corriger une ancienne copie de libtool fournie dans le code de Binutils.

Étape	Construction	Hôte	Cible	Action
1	pc	pc	lfs	Construire un compilateur

Étape	Construction	Hôte	Cible	Action
				croisé cc1 avec cc-pc sur pc.
2	pc	lfs	lfs	Construire un compilateur cc-lfs avec cc1 sur pc.
3	lfs	lfs	lfs	Reconstruire (et éventuellement tester) cc-lfs avec lui-même sur lfs.

Dans le tableau précédent, « sur pc » signifie que les commandes sont exécutées sur une machine qui utilise la distribution déjà installée. « Sur lfs » signifie que les commandes sont exécutées dans un environnement chroot.

Ce n'est pas la fin de l'histoire. Le langage C n'est pas seulement un compilateur, mais il définit aussi une bibliothèque standard. Dans ce livre, nous utilisons la bibliothèque C de GNU, appelée glibc (son alternative étant « musl »). Cette bibliothèque doit être compilée pour la machine LFS, c'est-à-dire à l'aide du compilateur croisé cc1. Mais le compilateur lui-même utilise une bibliothèque interne qui exécute des instructions complexes indisponibles dans le jeu d'instructions de l'assembleur. Cette bibliothèque interne, libgcc, doit être liée à la bibliothèque glibc pour fonctionner correctement ! De plus, la bibliothèque standard C++ (libstdc++) a aussi besoin d'être associée à glibc. La solution à ce problème consiste d'abord à construire une libgcc inférieure basée sur cc1, qui ne dispose pas de fonctionnalités avancées comme les threads et le traitement des exceptions, puis de construire glibc avec ce compilateur inférieur (glibc elle-même n'étant pas inférieure), puis de construire libstdc++. Cette bibliothèque ne dispose pas des fonctionnalités avancées de libgcc.

La conséquence du paragraphe précédent est que cc1 est incapable de construire une libstdc++ complètement fonctionnelle avec la libgcc dégradée, mais cc1 est le seul compilateur disponible pour construire les bibliothèques C/C++ à la deuxième étape. Comme indiqué, nous ne pouvons pas exécuter cc-lfs sur pc (la distribution hôte) car il peut nécessiter certaines bibliothèques, du code ou des données qui ne sont pas disponibles sur « la construction » (la distribution hôte). Ainsi, lorsque nous construisons la deuxième étape de gcc, nous remplaçons le chemin de recherche des bibliothèques pour se lier à libstdc++ de la libgcc nouvellement reconstruite au lieu de l'ancienne construction dégradée. Cela rend la libstdc++ reconstruite complètement fonctionnelle.

Dans le Chapitre 8, (ou « l'étape 3 »), tous les paquets nécessaires au système LFS sont construits. Même si vous avez déjà installé un paquet sur le système LFS dans un chapitre précédent, vous devrez reconstruire le paquet, à moins d'être certain qu'il n'est pas nécessaire. La stabilisation de ces paquets est la raison principale de leur reconstruction : si vous réinstallez un paquet LFS sur un système complet LFS, le contenu du paquet installé devrait être identique au contenu de ce paquet installé dans le Chapitre 8. Les paquets temporaires installés dans le Chapitre 6 ou le Chapitre 7 ne sont pas concernés, car certaines fonctionnalités facultatives sont désactivées à cause de dépendances manquantes ou du « mode de compilation croisée » . De plus, en reconstruisant les paquets, vous permettez l'exécution de la suite de tests.

Détails supplémentaires de procédure

Le compilateur croisé sera installé dans un répertoire `$LFS/tools` séparé, puisqu'il ne fera pas partie du système final.

Binutils est installé en premier parce que la commande **configure** de gcc et glibc effectuent des tests de fonctionnalités sur l'assembleur et l'éditeur de liens pour déterminer quelles fonctionnalités logicielles activer ou désactiver. Cette installation est plus importante que ce que vous pouvez penser. Un gcc ou une glibc mal configurés peuvent casser la chaîne de compilation, et l'impact d'une telle configuration ne se verrait qu'à la fin de la construction de la distribution complète. La suite de tests indiquera généralement cette erreur avant que vous n'ayez trop avancé.

Binutils installe son assembleur et son éditeur de liens à deux emplacements, `$LFS/tools/bin` et `$LFS/tools/$LFS_TGT/bin`. Les outils situés à un emplacement sont liés à l'autre par un lien en dur. L'ordre de recherche des bibliothèques est un aspect important de l'éditeur de liens. Vous pouvez obtenir des informations détaillées à partir de la commande **ld** en lui passant l'option *`--verbose`*. Par exemple, la commande **ld --verbose | grep SEARCH** affichera les chemins de recherche actuels et leur ordre. Cet exemple peut être exécuté en mode lecture par l'utilisateur `lfs`. Si vous revenez plus tard sur cette page, remplacez la commande **$LFS_TGT-ld** par **ld**.

Le prochain paquet installé est gcc. Voici un exemple de ce qui peut s'afficher pendant l'exécution de la commande **configure** :

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

Il s'agit d'un paquet important pour les raisons mentionnées plus haut. C'est également la preuve que le script configure de gcc ne parcourt pas les répertoires du PATH pour trouver quels outils utiliser. Cependant, lors de l'exécution normale de la commande **gcc**, les mêmes chemins de recherche ne sont pas forcément utilisés. Pour trouver quel éditeur de liens standard **gcc** utilise, exécutez la commande **$LFS_TGT-gcc -print-prog-name=ld**. À nouveau, retirez la commande **$LFS_TGT-** si vous revenez ici plus tard.

Vous pouvez obtenir des informations détaillées grâce à la commande **gcc** en lui passant l'option *`-v`* lors de la compilation d'un programme. Par exemple, la commande **$LFS_TGT-gcc -v *`example.c`*** (sans **$LFS_TGT-** si vous revenez plus tard) affichera des informations détaillées sur les phases de préprocesseur, de compilation et d'assemblage, ainsi que les chemins de recherche de **gcc** pour les en-têtes inclus et leur ordre.

L'installation suivante concerne les en-têtes nettoyés de l'API de Linux. Ils permettent à la bibliothèque standard C (glibc) d'interagir avec les fonctionnalités fournies par le noyau Linux.

Ensuite vient glibc. C'est le premier paquet que nous compilons de manière croisée. Nous utilisons l'option *`--host=$LFS_TGT`* pour faire utiliser ces outils préfixés par `$LFS_TGT` au système de construction, et l'option *`--build=$(../scripts/config.guess)`* pour activer le « mode de compilation croisée » comme indiqué. La variable `DESTDIR` est utilisée pour forcer l'installation dans le système de fichiers LFS.

Comme indiqué précédemment, la bibliothèque standard C++ est ensuite compilée, suivie dans le Chapitre 6 par les autres programmes qui nécessitent une compilation croisée en raison des dépendances circulaires qu'ils cassent lors de la construction. Les étapes pour ces paquets sont similaires aux étapes pour glibc.

À la fin du Chapitre 6, le compilateur LFS natif est installé. binutils-pass2 est construit en premier, avec le même répertoire d'installation `DESTDIR` que les autres programmes, puis la deuxième passe de gcc est construite sans les bibliothèques inutiles.

À l'entrée de l'environnement chroot dans le Chapitre 7, les programmes nécessaires au bon fonctionnement de la chaîne de compilation sont installés de manière temporaire. À partir de là, la chaîne de construction de base est auto-suffisante et auto-hébergée. Dans le Chapitre 8, vous construirez, testerez et installerez les versions finales de tous les paquets nécessaires au bon fonctionnement du système complet.

Instructions générales de compilation

Attention

Pendant un cycle de développement de LFS, les instructions du livre sont souvent modifiées pour s'adapter aux mises à jour de paquets ou pour profiter de nouvelles fonctionnalités des paquets mis à jour. Mélanger les instructions de plusieurs versions du livre peut causer des problèmes complexes. Ce genre de problème est en général le résultat de la réutilisation d'un script créé pour une version précédente de LFS. Une telle réutilisation est fortement découragée. Si vous réutilisez des scripts d'une version LFS précédente pour n'importe quelle raison, vous devrez faire attention à mettre à jour les scripts pour qu'ils correspondent à la version actuelle du livre LFS.

Voici des informations que vous devriez connaitre pour la construction des paquets :

- Plusieurs paquets sont corrigés avant d'être compilés, mais seulement dans le cas où la correction est nécessaire pour résoudre un problème. Souvent, le correctif est nécessaire à la fois dans ce chapitre et dans les suivants, mais quelques fois lorsque le même paquet est construit plus d'une fois, le correctif n'est pas immédiatement nécessaire. 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, des messages d'avertissement défileront sur votre écran. Ceci est normal et peut être ignoré sans danger. Ces messages d'avertissement concernent généralement une utilisation obsolète, mais pas invalide, de la syntaxe de C ou de C++. Les standards C changent assez souvent et certains paquets n'ont pas encore été mis à jour. Ce n'est pas un véritable problème mais cela provoque les messages.
- Vérifiez une dernière fois que la variable d'environnement `LFS` est configurée correctement :

```
echo $LFS
```

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

- Enfin, deux points importants doivent être précisés :

Important

Les instructions de construction supposent que vous avez défini correctement les Prérequis du système hôte, y compris les liens symboliques :

- **bash** est le shell utilisé.
- **sh** est un lien symbolique vers **bash**.
- **/usr/bin/awk** est un lien symbolique vers **gawk**.
- **/usr/bin/yacc** est un lien symbolique vers **bison** ou un petit script qui exécute bison.

Important

Voici un résumé du processus de construction.

1. Mettez tous les codes sources et les correctifs dans un répertoire qui sera accessible depuis l'environnement chroot, tel que `/mnt/lfs/sources/`.
2. Déplacez-vous vers le répertoire `/mnt/lfs/sources`.
3. Pour chaque paquet :
   a. En utilisant le programme **tar**, décompressez le paquet à construire. Dans les chapitres Chapitre 5 et Chapitre 6, assurez-vous d'être l'utilisateur *lfs* lors de l'extraction du paquet.

      N'utilisez aucune autre méthode que la commande **tar** pour extraire le code source. En particulier, l'utilisation de **cp -R** pour copier l'arborescence ailleurs peut détruire'horodatage de l'arborescence des sources et résulter en un échec à la construction.
   b. Allez dans le répertoire créé lorsque le paquet a été décompressé.
   c. Suivez les instructions pour construire le paquet.
   d. Revenez au répertoire des codes sources lorsque la construction est terminée.
   e. Supprimez le répertoire source que vous avez extrait sauf instruction contraire.

Chapitre 5. Compilation d'une chaîne d'outils croisée

5.1. Introduction

Ce chapitre montre comment construire un compilateur croisé et ses outils associés. Bien qu'ici la compilation croisée soit fausse, le principe est le même que pour une vraie compilation croisée.

Les programmes compilés dans ce chapitre vont être installés dans le répertoire `$LFS/tools` de façon à les garder séparés des fichiers installés dans les chapitres suivants. Les bibliothèques en revanche, sont installées à leur emplacement final, comme elles appartiennent au système que nous voulons construire.

5.2. Binutils-2.45 — Passe 1

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

Temps de construction approximatif: 1 SBU
Espace disque requis: 678 Mo

5.2.1. Installation de Binutils croisé

Note

Revenez en arrière et relisez les remarques de la section Instructions générales de compilation. La compréhension des remarques notées importantes vous évitera beaucoup de problèmes plus tard.

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

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

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

Note

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

Maintenant, préparez la compilation de Binutils :

```
../configure --prefix=$LFS/tools \
             --with-sysroot=$LFS \
             --target=$LFS_TGT   \
             --disable-nls       \
             --enable-gprofng=no \
             --disable-werror    \
             --enable-new-dtags  \
             --enable-default-hash-style=gnu
```

Voici la signification des options de configuration :

Note

Contrairement aux autres paquets, toutes les options listées plus bas n'apparaissent pas forcément quand on exécute **./configure --help**. Par exemple, pour trouver l'option `--with-sysroot`, vous devez exécuter **ld/configure --help**. Toutes les options peuvent être listées d'un coup avec **./configure --help=recursive**.

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

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

`--target=$LFS_TGT`
Vu que la description de la machine dans la variable `LFS_TGT` est légèrement différente de la valeur renvoyée par le script **config.guess**, ce paramètre va dire au script **configure** d'ajuster le système de construction de binutils pour la construction d'un éditeur de lien croisé.

`--disable-nls`
Ceci désactive l'internationalisation (i18n), car ce n'est pas nécessaire pour des outils temporaires.

`--enable-gprofng=no`
Ceci désactive la construction de grofng qui n'est pas requis pour les outils temporaires.

`--disable-werror`
Ceci empêche la compilation de s'arrêter lorsqu'interviennent des événements comme des avertissements du compilateur du système hôte.

`--enable-new-dtags`
Cela permet à l'éditeur de liens d'utiliser le tag « runpath » pour intégrer des chemins de recherche de bibliothèques dans les exécutables et les bibliothèques partagées, au lieu du tag « rpath » traditionnel. Cela facilite le débogage des exécutables liés dynamiquement et contourne des problèmes potentiels dans la suite de tests de certains paquets.

`--enable-default-hash-style=gnu`
Par défaut, l'éditeur de liens génère à la fois une table de hash GNU et la table de hash ELF classique pour les bibliothèques partagées et les exécutables liés dynamiquement. Les tables de hash ne sont conçues que pour que l'éditeur de liens puisse rechercher les symboles. Sur LFS l'éditeur de liens dynamique (fournit par le paquet Glibc) utilisera toujours la table de hash GNU qui est plus rapide. La table de hash ELF classique est donc absolument inutile. Cette option ne fait générer à l'éditeur de liens que la table de hash GNU par défaut, pour éviter de perdre du temps à générer la table de hash ELF classique lors de la construction des paquets ou de perdre de l'espace disque à la stocker.

Continuez avec la compilation du paquet :

```
make
```

Installez le paquet :

```
make install
```

Les détails sur ce paquet sont disponibles dans Section 8.20.2, « Contenu de Binutils. »

5.3. GCC-15.2.0 — Passe 1

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

Temps de construction approximatif: 3,8 SBU

Espace disque requis: 5,4 Go

5.3.1. Installation de GCC croisé

GCC requiert les paquets GMP, MPFR et MPC. Comme il se peut que ces paquets ne soient pas inclus dans votre distribution hôte, ils vont être compilés en même temps que GCC. Déballez chaque paquet dans le répertoire des sources de GCC et renommez les répertoires ainsi créés pour que les procédures de construction de GCC les utilisent automatiquement :

Note

Beaucoup d'incompréhensions existent concernant ce chapitre. Les procédures sont les mêmes que celles de tous les autres chapitres, comme expliqué plus tôt (Instructions de construction des paquets). Extrayez d'abord l'archive de gcc-15.2.0 du répertoire des sources puis rendez-vous dans le répertoire créé. C'est seulement là que vous devriez suivre les instructions ci-dessous.

```
tar -xf ../mpfr-4.2.2.tar.xz
mv -v mpfr-4.2.2 mpfr
tar -xf ../gmp-6.3.0.tar.xz
mv -v gmp-6.3.0 gmp
tar -xf ../mpc-1.3.1.tar.gz
mv -v mpc-1.3.1 mpc
```

Sur les systèmes x86_64, définissez « lib » comme nom de répertoire par défaut pour les bibliothèques 64 bits :

```
case $(uname -m) in
  x86_64)
    sed -e '/m64=/s/lib64/lib/' \
        -i.orig gcc/config/i386/t-linux64
 ;;
esac
```

Note

Cet exemple montre comment utiliser le paramètre *-i.orig*. Il fait copier le fichier `t-linux64` à **sed** vers `t-linux64.orig`, puis modifier le fichier original `t-linux64` directement. Ainsi, vous pouvez exécuter **diff -u gcc/config/i386/t-linux64{.orig,}** pour voir les changements effectués par la commande **sed** après coup. Nous utiliserons simplement *-i* (qui modifie seulement le fichier original directement sans le copier avant) pour tous les autres paquets de ce libre, mais vous pouvez utiliser *-i.orig* dans tous les cas où vous voudrez garder une copie de l'original.

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

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

Préparez la compilation de GCC :

```
../configure                   \
    --target=$LFS_TGT          \
    --prefix=$LFS/tools        \
    --with-glibc-version=2.42 \
    --with-sysroot=$LFS        \
    --with-newlib              \
    --without-headers          \
    --enable-default-pie       \
    --enable-default-ssp       \
    --disable-nls              \
    --disable-shared           \
    --disable-multilib         \
    --disable-threads          \
    --disable-libatomic        \
    --disable-libgomp          \
    --disable-libquadmath      \
    --disable-libssp           \
    --disable-libvtv           \
    --disable-libstdcxx        \
    --enable-languages=c,c++
```

Voici la signification des options de configuration :

`--with-glibc-version=2.42`
Cette option spécifie la version de Glibc qui sera utilisée sur la cible. Ce n'est pas lié à la libc de la distribution hôte parce que tout ce qui est compilé par GCC passe 1 sera lancé dans l'environnement chroot, qui est isolé de la libc de la distribution hôte.

`--with-newlib`
Vu qu'aucune bibliothèque C fonctionnelle n'est encore disponible, ceci garantit que la constante inhibit_libc soit définie lors de la construction de libgcc. Cela empêche la compilation d'un code exigeant la prise en charge de la libc.

`--without-headers`
Lors de la compilation d'un compilateur croisé complet, GCC exige des en-têtes standards compatibles avec le système cible. Pour nos objectifs, ces en-têtes ne seront pas nécessaires. Ce paramètre empêche GCC de les chercher.

`--enable-default-pie et --enable-default-ssp`
Ces paramètres permettent à GCC de compiler des programmes avec des fonctionnalités de durcissement (plus d'information dans remarque sur PIE et SSP au chapitre 8) par défaut. Elles ne sont pas vraiment requises à ce niveau, puisque le compilateur ne sera utilisé que pour produire les exécutables temporaires. Mais il est plus propre d'avoir des paquets temporaires aussi proches que possible des paquets finals.

`--disable-shared`
Ce paramètre oblige GCC à lier ses bibliothèques internes de manière statique. On procède ainsi parce que les bibliothèques partagées requièrent Glibc, qui n'est pas encore installé sur le système cible.

`--disable-multilib`
Sur du x86_64, LFS ne prend pas en charge une configuration multilib. Ce paramètre n'a pas d'importance pour x86.

`--disable-threads, --disable-libatomic, --disable-libgomp, --disable-libquadmath, --disable-libssp, --disable-libvtv, --disable-libstdcxx`
Ces paramètres désactivent la prise en charge de threading, libatomic, libgomp, libquadmath, libssp, libvtv et de la bibliothèque standard C++. La compilation de ces fonctions peut échouer lors de la construction d'un compilateur croisé et celles-ci sont inutiles pour la compilation croisée de la libc temporaire.

`--enable-languages=c,c++`
Cette option nous assure que seuls les compilateurs C et C++ seront construits. Ce sont les seuls langages actuellement nécessaires.

Compilez GCC en lançant :

```
make
```

Installez le paquet :

```
make install
```

Cette construction de GCC a installé quelques en-têtes internes au système. Normalement l'un d'entre eux, `limits.h`, inclurait à son tour l'en-tête `limits.h` du système, dans ce cas `$LFS/usr/include/limits.h`. Cependant, au moment de cette construction de GCC, `$LFS/usr/include/limits.h` n'existe pas, donc l'en-tête interne qui vient d'être construit est un fichier partiel, auto-contenu et n'inclus pas les fonctionnalités étendues de l'en-tête système. Cela est suffisant pour construire Glibc, mais l'en-tête interne complet sera requis plus tard. Créez une version complète de l'en-tête interne avec une commande identique à ce que le système de construction de GCC fait dans des circonstances normales :

Note

La commande ci-dessous montre un exemple de substitutions de commande imbriquées de deux manières : les backquotes et une construction `$()`. Elle pourrait être réécrite pour utiliser la même méthode pour les deux substitutions, mais elles sont montrées de cette manière pour montrer comment elles peuvent être mélangées. En général la méthode `$()` est préférable.

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

Les détails sur ce paquet sont disponibles dans Section 8.29.2, « Contenu de GCC. »

5.4. Linux-6.16.1 API Headers

Les en-têtes de l'API du noyau Linux (dans linux-6.16.1.tar.xz) expose les API du noyau à Glibc.

Temps de construction approximatif: moins de 0,1 SBU
Espace disque requis: 1,7 Go

5.4.1. Installation de Linux API Headers

Le noyau linux a besoin de montrer une interface de programmation de l'application (Application Programming Interface, API) à utiliser (Glibc dans LFS). Cela est possible en nettoyant certains fichiers d'en-tête C qui sont laissés dans le paquet des sources du noyau Linux.

Assurez-vous qu'il n'y a pas de vieux fichiers embarqués dans le paquet :

```
make mrproper
```

Maintenant extrayez les en-têtes publics du noyau depuis les sources. La cible make recommandée « headers_install » ne peut pas être utilisée car elle requiert rsync, qui n'est pas forcément disponible. On place les en-têtes dans `./usr` puis on les copie vers l'emplacement requis.

```
make headers
find usr/include -type f ! -name '*.h' -delete
cp -rv usr/include $LFS/usr
```

5.4.2. Contenu des en-têtes de l'API du noyau Linux

En-têtes installés: /usr/include/asm/*.h, /usr/include/asm-generic/*.h, /usr/include/drm/*.h, /usr/include/linux/*.h, /usr/include/misc/*.h, /usr/include/mtd/*.h, /usr/include/rdma/*.h, /usr/include/scsi/*.h, /usr/include/sound/*.h, /usr/include/video/*.h et /usr/include/xen/*.h
Répertoires installés: /usr/include/asm, /usr/include/asm-generic, /usr/include/drm, /usr/include/linux, /usr/include/misc, /usr/include/mtd, /usr/include/rdma, /usr/include/scsi, /usr/include/sound, /usr/include/video et /usr/include/xen

Descriptions courtes

`/usr/include/asm/*.h`	Les en-têtes ASM de l'API de Linux
`/usr/include/asm-generic/*.h`	Les en-têtes ASM génériques de l'API de Linux
`/usr/include/drm/*.h`	Les en-têtes DRM de l'API de Linux
`/usr/include/linux/*.h`	Les en-têtes linux de l'API de Linux
`/usr/include/misc/*.h`	Des en-têtes diverses de l'API de Linux
`/usr/include/mtd/*.h`	Les en-têtes MTD de l'API de Linux
`/usr/include/rdma/*.h`	Les en-têtes RDMA de l'API de Linux
`/usr/include/scsi/*.h`	Les en-têtes SCSI de l'API de Linux
`/usr/include/sound/*.h`	Les en-têtes son de l'API de Linux
`/usr/include/video/*.h`	Les en-têtes vidéo de l'API de Linux
`/usr/include/xen/*.h`	Les en-têtes Xen de l'API de Linux

5.5. Glibc-2.42

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

Temps de construction approximatif: 1,4 SBU
Espace disque requis: 870 Mo

5.5.1. Installation de Glibc

Tout d'abord, créez un lien symbolique pour respecter le LSB. De plus, pour x86_64, créez un lien symbolique de compatibilité requis pour le bon fonctionnement du chargeur de bibliothèques dynamiques :

```
case $(uname -m) in
    i?86)   ln -sfv ld-linux.so.2 $LFS/lib/ld-lsb.so.3
    ;;
    x86_64) ln -sfv ../lib/ld-linux-x86-64.so.2 $LFS/lib64
            ln -sfv ../lib/ld-linux-x86-64.so.2 $LFS/lib64/ld-lsb-x86-64.so.3
    ;;
esac
```

Note

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

Certains programmes de Glibc utilisent le répertoire `/var/db` qui ne respecte pas le FHS pour stocker leurs données d'exécution. Appliquez le correctif suivant pour que ces programmes stockent leurs données d'exécution aux emplacements indiqués par le FHS :

```
patch -Np1 -i ../glibc-2.42-fhs-1.patch
```

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

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

Assurez-vous que les utilitaires **ldconfig** et **sln** sont installés dans `/usr/sbin` :

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

Ensuite, préparez la compilation de Glibc :

```
../configure                             \
      --prefix=/usr                      \
      --host=$LFS_TGT                    \
      --build=$(../scripts/config.guess) \
      --disable-nscd                     \
      libc_cv_slibdir=/usr/lib           \
      --enable-kernel=5.4
```

Voici la signification des options de configuration :

`--host=$LFS_TGT, --build=$(../scripts/config.guess)`
L'effet combiné de ces commutateurs est que le système de construction de Glibc se configure pour se compiler de manière croisée en utilisant l'éditeur de liens croisé et le compilateur croisé dans `$LFS/tools`.

`--enable-kernel=5.4`
Ceci indique à Glibc de compiler la bibliothèque en prenant en charge les noyaux 5.4 et les noyaux Linux ultérieurs. Les contournements pour des noyaux plus anciens ne sont pas activés.

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

--disable-nscd
: Ne pas construire le démon de cache de service de nom qui n'est plus utilisé.

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

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

Le programme **msgfmt**, manquant ou incompatible, ne pose généralement pas de problème. Ce programme **msgfmt** fait partie du paquet Gettext que la distribution hôte devrait fournir.

Note

Il a été rapporté que ce paquet pouvait ne pas fonctionner en construisant avec un « make parallèle ». Si cela arrive, relancez la commande make avec l'option `-j1`.

Compilez le paquet :

```
make
```

Installez le paquet :

Avertissement

Si `LFS` n'est pas correctement configurée, et si malgré les recommandations vous construisez en `root`, la commande suivante installera la Glibc nouvellement construite sur votre système hôte, ce qui le rendra presque sûrement inutilisable. Alors assurez-vous que l'environnement est correctement initialisé et que vous n'êtes pas `root` avant de lancer la commande suivante.

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

Voici la signification de l'option de make install ::

DESTDIR=$LFS
: La variable make `DESTDIR` est utilisée par presque tous les paquets pour définir l'emplacement où le paquet sera installé. Si elle n'est pas configurée, elle renvoie par défaut à la racine (`/`). Ici, nous spécifions que le paquet doit être installé dans `$LFS`, qui deviendra la racine après le Section 7.4, « Entrer dans l'environnement chroot. »

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

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

Maintenant que notre chaîne d'outils croisée est en place, il est important de s'assurer que la compilation et l'édition des liens fonctionnent correctement. On peut s'en assurer en effectuant quelques vérifications :

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

Il ne devrait pas y avoir d'erreurs et la sortie de la dernière commande aura cette forme (en permettant au nom de l'éditeur des liens de différer en fonction de la plateforme) :

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

Remarquez que ce chemin ne devrait pas contenir `/mnt/lfs` (ou la valeur de la variable `LFS` si vous en utilisez une autre). Le chemin est résolu lorsque le programme compilé est exécuté, et cela ne devrait arriver qu'après être entré dans l'environnement chroot lorsque le noyau considérera que `$LFS` est le répertoire racine (`/`).

Maintenant, assurez-vous que vous êtes prêt à utiliser les bons fichiers :

```
grep -E -o "$LFS/lib.*/S?crt[1in].*succeeded" dummy.log
```

La sortie de la dernière commande devrait être :

```
/mnt/lfs/lib/../lib/Scrt1.o succeeded
/mnt/lfs/lib/../lib/crti.o succeeded
/mnt/lfs/lib/../lib/crtn.o succeeded
```

Vérifiez que le compilateur recherche les bons fichiers d'en-têtes :

```
grep -B3 "^ $LFS/usr/include" dummy.log
```

Cette commande devrait renvoyer la sortie suivante :

```
#include <...> search starts here:
 /mnt/lfs/tools/lib/gcc/x86_64-lfs-linux-gnu/15.2.0/include
 /mnt/lfs/tools/lib/gcc/x86_64-lfs-linux-gnu/15.2.0/include-fixed
 /mnt/lfs/usr/include
```

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

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

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

Les références au chemins qui ont des composantes comme « -linux-gnu » devraient être ignorés, mais sinon la sortie de la dernière commande devrait être :

```
SEARCH_DIR("=/mnt/lfs/tools/x86_64-lfs-linux-gnu/lib64")
SEARCH_DIR("=/usr/local/lib64")
SEARCH_DIR("=/lib64")
SEARCH_DIR("=/usr/lib64")
SEARCH_DIR("=/mnt/lfs/tools/x86_64-lfs-linux-gnu/lib")
SEARCH_DIR("=/usr/local/lib")
SEARCH_DIR("=/lib")
SEARCH_DIR("=/usr/lib");
```

Un système 32 bits peut utiliser quelques autres répertoires, mais dans tous les cas le point important ici est que tous les chemins devraient commencer par un signe égal (=), qui sera remplacé par le répertoire sysroot que nous avons configuré pour l'éditeur des liens.

Ensuite assurez-vous que vous utilisez la bonne libc :

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

La sortie de la dernière commande devrait être :

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

Assurez-vous que GCC utilise le bon éditeur dynamique :

```
grep found dummy.log
```

La sortie de la dernière commande devrait être (avec éventuellement des différences spécifiques à votre plateforme dans le nom de l'éditeur dynamique) :

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

Si la sortie ne ressemble pas à celle montrée ci-dessus ou si elle n'est pas disponible du tout, cela signifie que quelque chose s'est vraiment mal passé. Enquêtez et répétez les étapes pour trouver où les problèmes se trouvent et corrigez-les. Tout problème doit être résolu avant de continuer le processus.

Une fois que tout fonctionne correctement, nettoyez les fichiers de test :

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

Note

La construction des paquets dans le prochain chapitre servira de test supplémentaire pour vérifier que l'ensemble d'outils a été construit correctement. Si la construction de certains paquets échoue, en particulier Binutils-pass2 ou GCC-pass2, c'est une indication que quelque chose s’est mal passé dans les installations précédentes de Binutils, GCC, ou Glibc.

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

5.6. Libstdc++ de GCC-15.2.0

Libstdc++ est la bibliothèque standard du C++. Elle est requise pour compiler du code C++ (une partie de GCC est écrit en C++) mais nous avons dû retarder son installation lorsqu'on a construit gcc-pass1, car elle dépend de Glibc, qui n'était pas encore disponible dans le répertoire cible.

Temps de construction approximatif: 0,2 SBU
Espace disque requis: 1,3 Go

5.6.1. Installation de Libstdc++ Cible

Note

Libstdc++ fait partie des sources de GCC. Vous devriez d'abord déballer l'archive tar de GCC et vous rendre dans le répertoire `gcc-15.2.0`.

Créez un répertoire de construction séparé pour Libstdc++ et rentrez-y :

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

Préparez la compilation de Libstdc++ :

```
../libstdc++-v3/configure      \
    --host=$LFS_TGT            \
    --build=$(../config.guess) \
    --prefix=/usr              \
    --disable-multilib         \
    --disable-nls              \
    --disable-libstdcxx-pch    \
    --with-gxx-include-dir=/tools/$LFS_TGT/include/c++/15.2.0
```

Voici la signification des options de configuration :

`--host=...`
Indique d'utiliser le compilateur croisé que nous venons tout juste de construire à la place de celui de `/usr/bin`.

`--disable-libstdcxx-pch`
Ce paramètre empêche l'installation des fichiers inclus pré-compilés, qui ne sont pas nécessaires pour l'instant.

`--with-gxx-include-dir=/tools/$LFS_TGT/include/c++/15.2.0`
Cela spécifie le répertoire d'installation des fichiers d'en-tête. Comme Libstdc++ est la bibliothèque standard du C++ dans LFS, ce répertoire doit correspondre à l'emplacement où le compilateur C++ (**$LFS_TGT-g++**) cherche les fichiers d'en-tête C++ standards. Dans une construction standard, cette information est automatiquement passée aux options **configure** de Libstdc++ à partir du répertoire de plus haut niveau. Dans notre cas, cette information doit être passée explicitement. Le compilateur C++ ajoutera le chemin sysroot `$LFS` (spécifié à la construction de GCC passe 1) au début du chemin de recherche des en-têtes, pour qu'il recherche effectivement dans `$LFS/tools/$LFS_TGT/include/c++/15.2.0`. La combinaison de la variable *`DESTDIR`* (dans la commande **make install** ci-dessous) et ce paramètre s'assure d'installer les en-têtes à cet emplacement.

Compilez Libstdc++ en exécutant :

```
make
```

Installez la bibliothèque :

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

Supprimez les fichiers d'archive libtool car ils sont dangereux pour la compilation croisée :

```
rm -v $LFS/usr/lib/lib{stdc++{,exp,fs},supc++}.la
```

Les détails sur ce paquet sont disponibles dans Section 8.29.2, « Contenu de GCC. »

Chapitre 6. Compilation croisée des outils temporaires

6.1. Introduction

Ce chapitre montre comme compiler les utilitaires de base de manière croisée en utilisant la chaîne d'outils croisée qui vient d'être construite. Ces utilitaires sont installés à leur emplacement final, mais ne peuvent pas encore être utilisés. Les tâches de base utilisent toujours les outils de l'hôte. Cependant, les bibliothèques installées sont utilisées à l'édition des liens.

Il sera possible d'utiliser les utilitaires au prochain chapitre après être entré dans l'environnement « chroot ». Mais tous les paquets construits dans le chapitre actuel devront être construits avant de faire cela. Donc, nous ne pouvons pas encore être indépendants du système hôte.

Encore une fois, rappelons qu'une valeur incorrecte de `LFS` et la construction en tant que `root` peuvent rendre votre ordinateur inutilisable. Ce chapitre doit être entièrement effectué en tant qu'utilisateur `lfs`, avec l'environnement décrit dans Section 4.4, « Configurer l'environnement. »

6.2. M4-1.4.20

Le paquet M4 contient un processeur de macros.

Temps de construction approximatif: 0,1 SBU
Espace disque requis: 38 Mo

6.2.1. Installation de M4

Préparez la compilation de M4 :

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

Compilez le paquet :

```
make
```

Installez le paquet :

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

Les détails sur ce paquet sont disponibles dans Section 8.13.2, « Contenu de M4. »

6.3. Ncurses-6.5-20250809

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

Temps de construction approximatif: 0,4 SBU
Espace disque requis: 54 Mo

6.3.1. Installation de Ncurses

Tout d'abord, exécutez les commandes suivantes pour construire le programme **tic** sur l'hôte de construction. Nous l'installons dans `$LFS/tools` pour qu'il soit trouvé par la variable `PATH` lorsque cela sera nécessaire :

```
mkdir build
pushd build
  ../configure --prefix=$LFS/tools AWK=gawk
  make -C include
  make -C progs tic
  install progs/tic $LFS/tools/bin
popd
```

Préparez la compilation de Ncurses :

```
./configure --prefix=/usr                \
            --host=$LFS_TGT              \
            --build=$(./config.guess)    \
            --mandir=/usr/share/man      \
            --with-manpage-format=normal \
            --with-shared                \
            --without-normal             \
            --with-cxx-shared            \
            --without-debug              \
            --without-ada                \
            --disable-stripping          \
            AWK=gawk
```

Voici la signification des nouvelles options de configure :

`--with-manpage-format=normal`
Cela évite que Ncurses n'installe les pages de manuel compressées, ce qui peut arriver si la distribution hôte elle-même a des pages de manuel compressées.

`--with-shared`
Cette option fait construire et installer les bibliothèques C partagées de Ncurses.

`--without-normal`
Cette option empêche Ncurses de construire et d'installer les bibliothèques C statiques.

`--without-debug`
Cette option empêche Ncurses de construire et d'installer les bibliothèques de débogage.

`--with-cxx-shared`
Cela fait construire et installer les liaisons C++ partagées de Ncurses. Cela l'empêche également de construire et d'installer les liaisons C++ statiques.

`--without-ada`
Cela s'assure que Ncurses ne construise pas la prise en charge du compilateur Ada qui peut être présent sur l'hôte mais qui ne sera pas disponible une fois dans l'environnement **chroot**.

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

AWK=gawk
Ce paramètre évite que le système de construction n'utilise le programme **mawk** de l'hôte. Certaines versions de **mawk** peuvent causer l'échec de la construction de ce paquet.

Compilez le paquet :

```
make
```

Installez le paquet :

```
make DESTDIR=$LFS install
ln -sv libncursesw.so $LFS/usr/lib/libncurses.so
sed -e 's/^#if.*XOPEN.*$/#if 1/' \
    -i $LFS/usr/include/curses.h
```

Voici la signification des options d'installation :

ln -sv libncursesw.so $LFS/usr/lib/libncurses.so
La bibliothèque `libncurses.so` est requise par quelques paquets que nous allons bientôt construire. Nous créons ce lien symbolique pour utiliser `libncursesw.so` à la place.

sed -e 's/^#if.*XOPEN.*$/#if 1/' ...
Le fichier d'en-tête `curses.h` contient les définitions de plusieurs structures de données de Ncurses. À cause des différentes définitions de macro du préprocesseur, deux ensembles de définitions de structures de données peuvent être utilisées : la définition 8-bits est compatible avec `libncurses.so` et la définition wide-character est compatible avec `libncursesw.so`. Comme nous utilisons `libncursesw.so` en remplacement de `libncurses.so`, modifiez le fichier d'en-tête pour qu'il utilise toujours le définition de la structure de données wide-character compatible avec `libncursesw.so`.

Les détails sur ce paquet sont disponibles dans Section 8.30.2, « Contenu de Ncurses. »

6.4. Bash-5.3

Le paquet Bash contient le Bourne-Again Shell.

Temps de construction approximatif: 0,2 SBU
Espace disque requis: 72 Mo

6.4.1. Installation de Bash

Préparez la compilation de Bash :

```
./configure --prefix=/usr                      \
            --build=$(sh support/config.guess) \
            --host=$LFS_TGT                    \
            --without-bash-malloc
```

Voici la signification des options de configuration :

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

Compilez le paquet :

```
make
```

Installez le paquet :

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

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

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

Les détails sur ce paquet sont disponibles dans Section 8.36.2, « Contenu de Bash. »

6.5. Coreutils-9.7

Le paquet Coreutils contient les utilitaires de base requis dans tous les systèmes d'exploitation.

Temps de construction approximatif: 0,3 SBU
Espace disque requis: 181 Mo

6.5.1. Installation de Coreutils

Préparez la compilation de Coreutils :

```
./configure --prefix=/usr                     \
            --host=$LFS_TGT                   \
            --build=$(build-aux/config.guess) \
            --enable-install-program=hostname \
            --enable-no-install-program=kill,uptime
```

Voici la signification des options de configuration :

`--enable-install-program=hostname`
Ceci fait que le binaire **hostname** sera compilé et installé – ceci est désactivé par défaut mais est requis par la suite de tests de Perl.

Compilez le paquet :

```
make
```

Installez le paquet :

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

Déplacez les programmes à leurs emplacements définitifs. Bien que ce ne soit pas nécessaire dans cet environnement temporaire, nous devons le faire parce que certains programmes codent la position des exécutables en dur :

```
mv -v $LFS/usr/bin/chroot              $LFS/usr/sbin
mkdir -pv $LFS/usr/share/man/man8
mv -v $LFS/usr/share/man/man1/chroot.1 $LFS/usr/share/man/man8/chroot.8
sed -i 's/"1"/"8"/'                    $LFS/usr/share/man/man8/chroot.8
```

Les détails sur ce paquet sont disponibles dans Section 8.59.2, « Contenu de Coreutils. »

6.6. Diffutils-3.12

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

Temps de construction approximatif: 0,1 SBU
Espace disque requis: 35 Mo

6.6.1. Installation de Diffutils

Préparez la compilation de Diffutils :

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

Voici la signification des options de configuration :

`gl_cv_func_strcasecmp_works=y`
Cette option spécifie le résultat d'un test pour `strcasecmp`. Le test nécessite d'exécuter un programme C compilé, et cela est impossible à cause de la compilation croisée car en général un programme compilé de manière croisée ne peut pas s'exécuter sur la distribution hôte. Normalement pour ce genre de test, le script **configure** utilisera une valeur par défaut pour la compilation croisée, mais la valeur par défaut pour ce test est absente et le script **configure** n'aura pas de valeur à utiliser et s'arrêtera avec une erreur. Les développeurs amont ont déjà corrigé le problème, mais pour appliquer le correctif il faut exécuter **autoconf**, ce que la distribution hôte n'a pas forcément. Nous spécifions donc simplement le résultat du test (`y` comme nous savons que la fonction `strcasecmp` de Glibc-2.42 fonctionne correctement). Ensuite, **configure** utilisera simplement cette valeur et passera le test.

Compilez le paquet :

```
make
```

Installez le paquet :

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

Les détails sur ce paquet sont disponibles dans Section 8.60.2, « Contenu de Diffutils. »

6.7. File-5.46

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

Temps de construction approximatif: 0,1 SBU
Espace disque requis: 42 Mo

6.7.1. Installation de File

La commande **file** sur l'hôte de construction doit être à la même version que celle que nous construisons pour créer le fichier de signature. Lancez les commandes suivantes pour créer une copie temporaire de la commande **file** :

```
mkdir build
pushd build
  ../configure --disable-bzlib      \
               --disable-libseccomp \
               --disable-xzlib      \
               --disable-zlib
  make
popd
```

Voici la signification de la nouvelle option de configuration :

`--disable-*`
Le script de configuration essaye d'utiliser certains paquets de la distribution hôte si les fichiers de bibliothèques correspondantes existent. Cela peut causer un échec à la construction si un fichier de bibliothèque existe, mais pas les fichiers d'en-têtes correspondants. Ces options évitent d'utiliser ces fonctionnalités inutiles de l'hôte.

Préparez la compilation de File :

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

Compilez le paquet :

```
make FILE_COMPILE=$(pwd)/build/src/file
```

Installez le paquet :

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

Supprimez le fichier d'archive libtool car il est dangereux pour la compilation croisée :

```
rm -v $LFS/usr/lib/libmagic.la
```

Les détails sur ce paquet sont disponibles dans Section 8.11.2, « Contenu de File. »

6.8. Findutils-4.10.0

Le paquet Findutils contient des programmes de recherche de fichiers. Ces programmes sont fournis pour parcourir tous les fichiers dans une hiérarchie de répertoires et pour créer, maintenir et parcourir 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). Findutils fournit également le programme **xargs**, qui peut être utilisé pour exécuter une commande spécifique sur chaque fichier sélectionné par la recherche.

Temps de construction approximatif: 0,2 SBU
Espace disque requis: 48 Mo

6.8.1. Installation de Findutils

Préparez la compilation de Findutils :

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

Compilez le paquet :

```
make
```

Installez le paquet :

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

Les détails sur ce paquet sont disponibles dans Section 8.62.2, « Contenu de Findutils. »

6.9. Gawk-5.3.2

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

Temps de construction approximatif: 0,1 SBU
Espace disque requis: 49 Mo

6.9.1. Installation de Gawk

Tout d'abord, assurez-vous que certains fichiers inutiles ne sont pas installés :

```
sed -i 's/extras//' Makefile.in
```

Préparez la compilation de Gawk :

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

Compilez le paquet :

```
make
```

Installez le paquet :

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

Les détails sur ce paquet sont disponibles dans Section 8.61.2, « Contenu de Gawk. »

6.10. Grep-3.12

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

Temps de construction approximatif: 0,1 SBU
Espace disque requis: 32 Mo

6.10.1. Installation de Grep

Préparez la compilation de Grep :

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

Compilez le paquet :

```
make
```

Installez le paquet :

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

Les détails sur ce paquet sont disponibles dans Section 8.35.2, « Contenu de Grep. »

6.11. Gzip-1.14

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

Temps de construction approximatif: 0,1 SBU
Espace disque requis: 12 Mo

6.11.1. Installation de Gzip

Préparez la compilation de Gzip :

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

Compilez le paquet :

```
make
```

Installez le paquet :

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

Les détails sur ce paquet sont disponibles dans Section 8.65.2, « Contenu de Gzip. »

6.12. Make-4.4.1

Le paquet Make contient un programme pour contrôler la génération d'exécutables et d'autres fichiers non-sources d'un paquet à partir des fichiers sources.

Temps de construction approximatif: moins de 0,1 SBU
Espace disque requis: 15 Mo

6.12.1. Installation de Make

Préparez la compilation de Make :

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

Compilez le paquet :

```
make
```

Installez le paquet :

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

Les détails sur ce paquet sont disponibles dans Section 8.69.2, « Contenu de Make. »

6.13. Patch-2.8

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

Temps de construction approximatif: 0,1 SBU
Espace disque requis: 14 Mo

6.13.1. Installation de Patch

Préparez la compilation de Patch :

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

Compilez le paquet :

```
make
```

Installez le paquet :

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

Les détails sur ce paquet sont disponibles dans Section 8.70.2, « Contenu de Patch. »

6.14. Sed-4.9

Le paquet Sed contient un éditeur de flux.

Temps de construction approximatif:	0,1 SBU
Espace disque requis:	21 Mo

6.14.1. Installation de Sed

Préparez la compilation de Sed :

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

Compilez le paquet :

```
make
```

Installez le paquet :

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

Les détails sur ce paquet sont disponibles dans Section 8.31.2, « Contenu de Sed. »

6.15. Tar-1.35

Le paquet Tar fournit la possibilité de créer des archives tar et effectuer diverses manipulations d'archives. Tar peut être utilisé sur des archives précédemment créées pour extraire des fichiers, ajouter des fichiers supplémentaires, mettre à jour ou lister les fichiers qui étaient déjà stockés.

Temps de construction approximatif: 0,1 SBU
Espace disque requis: 42 Mo

6.15.1. Installation de Tar

Préparez la compilation de Tar :

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

Compilez le paquet :

```
make
```

Installez le paquet :

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

Les détails sur ce paquet sont disponibles dans Section 8.71.2, « Contenu de Tar. »

6.16. Xz-5.8.1

Le paquet Xz contient des programmes de compression et de décompression de fichiers. Il offre les possibilités des formats lzma et des formats de compression récents. La compression de fichiers textes avec **xz** donne un meilleur pourcentage de compression qu'avec les commandes **gzip** ou **bzip2** traditionnelles.

Temps de construction approximatif: 0,1 SBU
Espace disque requis: 23 Mo

6.16.1. Installation de Xz

Préparez la compilation de Xz :

```
./configure --prefix=/usr                     \
            --host=$LFS_TGT                   \
            --build=$(build-aux/config.guess) \
            --disable-static                  \
            --docdir=/usr/share/doc/xz-5.8.1
```

Compilez le paquet :

```
make
```

Installez le paquet :

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

Supprimez le fichier d'archive libtool car il est dangereux pour la compilation croisée :

```
rm -v $LFS/usr/lib/liblzma.la
```

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

6.17. Binutils-2.45 — Passe 2

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

Temps de construction approximatif: 0,4 SBU
Espace disque requis: 548 Mo

6.17.1. Installation de Binutils

Le système de construction de Binutils se base sur une copie de libtool pour se lier à des bibliothèques statiques internes, mais les copies de libiberty et zlib fournies dans le paquet n'utilisent pas libtool. Cette incohérence peut faire que les binaires produits seront liés aux bibliothèques de l'hôte par erreur. Contournez ce problème :

```
sed '6031s/$add_dir//' -i ltmain.sh
```

Créez de nouveau un répertoire de construction séparé :

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

Préparez la compilation de Binutils :

```
../configure                   \
    --prefix=/usr              \
    --build=$(../config.guess) \
    --host=$LFS_TGT            \
    --disable-nls              \
    --enable-shared            \
    --enable-gprofng=no        \
    --disable-werror           \
    --enable-64-bit-bfd        \
    --enable-new-dtags         \
    --enable-default-hash-style=gnu
```

Voici la signification des nouvelles options de configure :

`--enable-shared`
Construit `libbfd` en tant que bibliothèque partagée.

`--enable-64-bit-bfd`
Active la prise en charge du 64 bits (sur les hôtes avec des tailles de mots plus petites). Cela n'est peut-être pas nécessaire sur les systèmes 64 bits, mais ça ne fait pas de mal.

Compilez le paquet :

```
make
```

Installez le paquet :

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

Supprimez les fichiers d'archive libtool car ils sont dangereux pour la compilation croisée et supprimez des bibliothèques statiques inutiles :

```
rm -v $LFS/usr/lib/lib{bfd,ctf,ctf-nobfd,opcodes,sframe}.{a,la}
```

Les détails sur ce paquet sont disponibles dans Section 8.20.2, « Contenu de Binutils. »

6.18. GCC-15.2.0 — Passe 2

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

Temps de construction approximatif: 4,5 SBU
Espace disque requis: 6,0 Go

6.18.1. Installation de GCC

Comme pour la première construction de GCC, les paquets GMP, MPFR et MPC sont requis. Déballez les archives et déplacez-les dans les répertoires avec le nom requis :

```
tar -xf ../mpfr-4.2.2.tar.xz
mv -v mpfr-4.2.2 mpfr
tar -xf ../gmp-6.3.0.tar.xz
mv -v gmp-6.3.0 gmp
tar -xf ../mpc-1.3.1.tar.gz
mv -v mpc-1.3.1 mpc
```

Si vous construisez sur x86_64, changez le nom du répertoire par défaut des bibliothèques 64 bits en « lib » :

```
case $(uname -m) in
  x86_64)
    sed -e '/m64=/s/lib64/lib/' \
        -i.orig gcc/config/i386/t-linux64
  ;;
esac
```

Remplacez les règles de construction des en-têtes de libgcc et libstdc++ pour permettre la construction de ces bibliothèques avec la prise en charge des threads POSIX :

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

Créez de nouveau un répertoire de construction séparé :

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

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

Maintenant, préparez la compilation de GCC :

```
../configure                   \
    --build=$(../config.guess) \
    --host=$LFS_TGT            \
    --target=$LFS_TGT          \
    --prefix=/usr              \
    --with-build-sysroot=$LFS  \
    --enable-default-pie       \
    --enable-default-ssp       \
    --disable-nls              \
    --disable-multilib         \
    --disable-libatomic        \
    --disable-libgomp          \
    --disable-libquadmath      \
    --disable-libsanitizer     \
    --disable-libssp           \
    --disable-libvtv           \
    --enable-languages=c,c++   \
    LDFLAGS_FOR_TARGET=-L$PWD/$LFS_TGT/libgcc
```

Voici la signification des nouvelles options de configure :

`--with-build-sysroot=$LFS`
Normalement, utiliser *`--host`* s'assure qu'un compilateur croisé est utilisé pour construire GCC, et ce compilateur sait qu'il doit chercher les en-têtes et les bibliothèques dans `$LFS`. Cependant, le système de construction de GCC utilise des outils supplémentaires qui ne connaissent pas cet emplacement. Ce paramètre est requis pour qu'ils trouvent les fichiers requis dans `$LFS` et non sur l'hôte.

`--target=$LFS_TGT`
Comme nous effectuons une compilation croisée de GCC, il n'est pas possible de construire les bibliothèques de la cible (`libgcc` et `libstdc++`) avec les binaires GCC compilés dans cette passe : ces binaires ne fonctionneront pas sur l'hôte. Le système de construction de GCC essayera d'utiliser les compilateurs C et C++ de l'hôte pour contourner cela par défaut. La construction des bibliothèques GCC de la cible avec une version différente de GCC n'est pas prise en charge, donc utiliser les compilateurs de l'hôte peut causer des échecs de construction. Ce paramètre s'assure de construire les bibliothèques avec GCC passe 1.

`LDFLAGS_FOR_TARGET=...`
Permet à `libstdc++` d'utiliser la `libgcc` en cours de construction dans cette passe, au lieu de la version précédemment construite dans gcc-pass1. La version précédente ne prend pas correctement en charge les exceptions C++ car elle a été construite sans la prise en charge de la libc.

`--disable-libsanitizer`
Désactive les bibliothèques d'assainissement GCC à l'exécution. Elles ne sont pas requises pour l'installation temporaire. Dans gcc-pass1 il était sous-entendu par *`--disable-libstdcxx`*, et maintenant nous pouvons le passer explicitement.

Compilez le paquet :

```
make
```

Installez le paquet :

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

Comme touche finale, créez un lien symbolique utilitaire. De nombreux programmes et scripts lancent **cc** au lieu de **gcc**, pour que les programmes restent génériques et donc utilisables sur n'importe quel type de système UNIX où le compilateur C de GNU n'est pas toujours installé. Lancer **cc** laisse l'administrateur système libre de décider quel compilateur C installer :

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

Les détails sur ce paquet sont disponibles dans Section 8.29.2, « Contenu de GCC. »

Chapitre 7. Entrée dans le chroot et construction des outils temporaires supplémentaires

7.1. Introduction

Ce chapitre indique comment construire les derniers éléments manquants du système temporaire, c'est-à-dire les outils requis pour construire les différents paquets. Maintenant que toutes les dépendances circulaires ont été résolues, nous pouvons utiliser un environnement « chroot » complètement isolé du système d'exploitation hôte (à l'exception du noyau en cours d'exécution) pour la construction.

Pour faire fonctionner correctement l'environnement isolé, il faut établir la communication avec le noyau à travers ce qu'on appelle les *Virtual Kernel File Systems*. Ceux-ci seront montés avant d'entrer dans l'environnement chroot. Pour vérifier qu'ils sont bien montés, exécutez la commande **findmnt**.

Jusqu'à la Section 7.4, « Entrer dans l'environnement chroot », les commandes doivent être exécutées en tant qu'utilisateur `root` avec la variable d'environnement `LFS`. Une fois entrées dans le chroot, toutes les commandes sont exécutées en tant qu'utilisateur `root`, heureusement sans avoir accès à l'OS de l'ordinateur sur lequel vous construisez LFS. Restez prudent malgré tout, car il est facile d'altérer l'entièreté du système LFS en exécutant de mauvaises commandes.

7.2. Changement du propriétaire

Note

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

actuellement, la hiérarchie complète des répertoires de `$LFS` appartient à l'utilisateur `lfs`, un utilisateur qui n'existe que sur le système hôte. Si les répertoires et les fichiers dans `$LFS` sont conservés ainsi, ils appartiendront à un ID utilisateur sans compte correspondant. C'est dangereux car un compte utilisateur créé plus tard pourrait se voir attribuer ce même ID utilisateur et être propriétaire de tous les fichiers du répertoire `$LFS`, ce qui exposerait alors ces fichiers à de possibles manipulations malveillantes.

Pour éviter ce problème, changez le propriétaire du répertoire `$LFS` pour l'attribuer à l'utilisateur `root` en exécutant la commande suivante :

```
chown --from lfs -R root:root $LFS/{usr,var,etc,tools}
case $(uname -m) in
  x86_64) chown --from lfs -R root:root $LFS/lib64#;;
esac
```

7.3. Préparer les systèmes de fichiers virtuels du noyau

Les applications qui tournent en espace utilisateur utilisent différents systèmes de fichiers créés par le noyau pour communiquer avec le noyau lui-même. Ces systèmes de fichiers sont virtuels du fait qu'ils n'utilisent aucun espace disque. Le contenu de ces systèmes de fichiers réside en mémoire. Ces systèmes de fichiers doivent être montés dans l'arborescence de $LFS pour que les applications puissent les trouver dans l'environnement chroot.

Commencez par créer les répertoires dans lesquels les systèmes de fichiers seront montés :

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

7.3.1. Monter et alimenter /dev

Lors d'un démarrage normal d'un système LFS, le noyau monte automatiquement le système de fichiers `devtmpfs` dans le répertoire `/dev` et crée des nœuds de périphériques sur ce système de fichiers virtuel pendant le processus de démarrage ou lorsqu'un périphérique est détecté ou qu'on tente d'y pour la première fois. Le démon udev permet de modifier le propriétaire, de gérer les permissions des nœuds de périphériques créés par le noyau, d'en créer de nouveaux ou de créer des liens symboliques afin de faciliter la tâche de maintenance de distribution ou d'administration système (voir le Section 9.3.2.2, « Création de nœuds de périphérique » pour plus de détails). Si le noyau hôte prend en charge `devtmpfs`, il est possible de monter un `devtmpfs` sous `$LFS/dev` et ainsi laisser le noyau le remplir.

Cependant, certains systèmes hôtes ne prennent pas en charge `devtmpfs`. Ces distributions hôtes utilisent plusieurs méthodes pour créer le contenu de `/dev`. La seule manière de remplir `$LFS/dev` indépendamment de l'hôte consiste à monter le répertoire `/dev` du système hôte avec l'option bind. Le montage avec --bind est un type spécial de montage qui vous permet de créer le miroir d'un répertoire ou d'un point de montage à un autre endroit. Pour ce faire, exécutez la commande suivante.

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

7.3.2. Monter les systèmes de fichiers virtuels du noyau

Montez maintenant les systèmes de fichiers virtuels du noyau restants :

```
mount -vt devpts devpts -o gid=5,mode=0620 $LFS/dev/pts
mount -vt proc proc $LFS/proc
mount -vt sysfs sysfs $LFS/sys
mount -vt tmpfs tmpfs $LFS/run
```

Voici la signification des options de montage pour devpts :

`gid=5`

Cela s'assure que tous les nœud de périphériques créés par devpts appartiennent au groupe 5. Il s'agit de l'identifiant qui sera utilisé plus tard pour le groupe `tty`. Nous utilisons l'identifiant du groupe au lieu d'un nom, car le système hôte pourrait utiliser un ID différent pour son groupe `tty`.

`mode=0620`

Cela s'assure que tous les nœuds de périphérique créés par devpts ont le mode 0620 (lisible et inscriptible par l'utilisateur, inscriptible par le groupe). Avec l'option précédente, cela s'assure que devpts créera des nœuds de périphérique qui respectent les prérequis de grantpt(), ce qui signifie que le binaire auxiliaire **pt_chown** Glibc (qui n'est pas installé par défaut) n'est pas nécessaire.

Dans certains systèmes hôtes, `/dev/shm` est un lien symbolique vers un répertoire, en général `/run/shm`. Le tmpfs / run a été monté plus tôt, donc, dans ce cas précis, vous aurez uniquement à créer un répertoire avec les bonnes permissions.

Sur d'autres systèmes hôtes, `/dev/shm` est un point de montage pour un tmpfs. Dans ce cas, le montage de /dev ci-dessus créera le répertoire /dev/shm dans l'environnement chroot. En parallèle, il faudra alors créer explicitement un tmpfs :

```
if [ -h $LFS/dev/shm ]; then
  install -v -d -m 1777 $LFS$(realpath /dev/shm)
else
  mount -vt tmpfs -o nosuid,nodev tmpfs $LFS/dev/shm
fi
```

7.4. Entrer dans l'environnement chroot

Maintenant que tous les paquets requis pour construire le reste des outils nécessaires sont sur le système, il est temps d'entrer dans l'environnement chroot pour finir l'installation des outils temporaires. Nous utiliserons aussi cet environnement pour l'installation du système final. En tant que `root`, lancez la commande suivante pour entrer dans cet environnement qui, pour le moment, contient seulement les outils temporaires :

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

Si vous ne voulez pas utiliser tous les cœurs logiques disponibles, remplacez *`$(nproc)`* par le nombre de cœurs logiques que vous voulez utiliser pour construire les paquets de ce chapitre et des chapitres suivants. Les suites de test de certains paquets (notamment AUtoconf, Libtool et Tar) dans Chapitre 8 ne sont pas affectées par `MAKEFLAGS`, elles utilisent une variable d'environnement `TESTSUITEFLAGS` à la place. Nous l'indiquons également ici pour exécuter ces tests avec plusieurs cœurs.

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

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. La commande **chroot** exécute l'interpréteur de commande Bash avec le répertoire racine (`/`) correspondant à `$LFS`.

Remarquez que `/tools/bin` n'est pas dans le `PATH`. Ceci signifie que la chaîne d'outils croisée ne sera plus utilisée.

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

Note

Il est important que toutes les commandes au sein du reste de ce chapitre et des chapitres suivants soient exécutées à l'intérieur de l'environnement chroot. Si vous devez quitter cet environnement pour une quelconque raison (un redémarrage par exemple), assurez-vous que les systèmes de fichiers virtuels du noyau sont montés comme expliqué dans Section 7.3.1, « Monter et alimenter /dev » et Section 7.3.2, « Monter les systèmes de fichiers virtuels du noyau » et entrez de nouveau dans le chroot avant de continuer l'installation.

7.5. Création des répertoires

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

Note

Certains des répertoires suivants ont déjà été créés plus tôt avec des instructions explicites ou lors de l'installation de certains paquets. Ils sont répétés ici par souci d'exhaustivité.

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

```
mkdir -pv /{boot,home,mnt,opt,srv}
```

Créez l'ensemble de sous-répertoires requis sous la racine en exécutant les commandes suivantes :

```
mkdir -pv /etc/{opt,sysconfig}
mkdir -pv /lib/firmware
mkdir -pv /media/{floppy,cdrom}
mkdir -pv /usr/{,local/}{include,src}
mkdir -pv /usr/lib/locale
mkdir -pv /usr/local/{bin,lib,sbin}
mkdir -pv /usr/{,local/}share/{color,dict,doc,info,locale,man}
mkdir -pv /usr/{,local/}share/{misc,terminfo,zoneinfo}
mkdir -pv /usr/{,local/}share/man/man{1..8}
mkdir -pv /var/{cache,local,log,mail,opt,spool}
mkdir -pv /var/lib/{color,misc,locate}

ln -sfv /run /var/run
ln -sfv /run/lock /var/lock

install -dv -m 0750 /root
install -dv -m 1777 /tmp /var/tmp
```

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

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

7.5.1. Remarques à propos de la conformité FHS

L'arborescence des répertoires est basée sur le standard FHS (*Filesystem Hierarchy Standard*, hiérarchie standard du système de fichiers, disponible sur *https://refspecs.linuxfoundation.org/fhs.shtml*). Le FHS mentionne aussi l'existence de quelques répertoires comme `/usr/local/games` et `/usr/share/games`. Nous créons seulement les répertoires nécessaires. Néanmoins, n'hésitez pas à créer ces répertoires.

Avertissement

La FHS ne dicte pas l'existence du répertoire `/usr/lib64` et les éditeurs de LFS ont décidé de ne pas l'utiliser. Pour que les instructions dans LFS et BLFS fonctionnent correctement, il est nécessaire que ce répertoire n'existe pas. De temps en temps vous devriez vérifier qu'il n'existe pas, car il est facile de le créer accidentellement et cela cassera probablement votre système.

7.6. Création des fichiers et des liens symboliques essentiels

Historiquement, Linux gardait une liste des systèmes de fichiers montés dans le fichier `/etc/mtab`. Les noyaux modernes gèrent cette liste en interne et la proposent à l'utilisateur via le système de fichiers `/proc`. Afin de satisfaire les outils qui s'attendent à la présence de `/etc/mtab`, créez le lien symbolique suivant :

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

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

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

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

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

```
cat > /etc/passwd << "EOF"
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/dev/null:/usr/bin/false
daemon:x:6:6:Daemon User:/dev/null:/usr/bin/false
messagebus:x:18:18:D-Bus Message Daemon User:/run/dbus:/usr/bin/false
uuidd:x:80:80:UUID Generation Daemon User:/dev/null:/usr/bin/false
nobody:x:65534:65534:Unprivileged User:/dev/null:/usr/bin/false
EOF
```

Le mot de passe réel pour `root` sera paramétré plus tard.

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

```
cat > /etc/group << "EOF"
root:x:0:
bin:x:1:daemon
sys:x:2:
kmem:x:3:
tape:x:4:
tty:x:5:
daemon:x:6:
floppy:x:7:
disk:x:8:
lp:x:9:
dialout:x:10:
audio:x:11:
video:x:12:
utmp:x:13:
cdrom:x:15:
adm:x:16:
messagebus:x:18:
input:x:24:
mail:x:34:
kvm:x:61:
uuidd:x:80:
wheel:x:97:
users:x:999:
nogroup:x:65534:
EOF
```

Les groupes créés ne font partie d'aucun standard, ce sont des groupes décidés d'un part en fonction des besoins de la configuration de Udev dans le chapitre 9, et d'autre part par la convention usuelle d'un certain nombre de distributions Linux existantes. En outre, certaines suites de tests s'appuient sur des groupes et des utilisateurs spécifiques. La base Linux standard (Linux Standard Base ou LSB, disponible sur *https://refspecs.linuxfoundation.org/lsb.shtml*) ne recommande uniquement en plus de la présence d'un groupe `root` accompagné d'un ID de groupe (GID) de 0, qu'un groupe `bin` soit accompagné d'un GID de 1. Le GID 5 est souvent utilisé pour le groupe `tty` et le numéro 5 est aussi utilisé dans `/etc/fstab` pour le système de fichiers `devpts`. Tous les autres noms de groupe et GID peuvent être librement choisis par l'administrateur du système puisque les programmes bien écrits ne dépendent pas des numéros GID, mais utilisent plutôt le nom du groupe.

L'ID 65534 est utilisé par le noyau pour NFS et les espaces de noms séparés pour les utilisateurs et les groupes non projetés (ils existent sur le serveur NFS ou dans l'espace de nom parent, mais « n'existent pas » sur la machine locale ou dans l'espace de nom séparé). Nous assignons l'utilisateur `nobody` et le groupe `nogroup` pour éviter d'avoir un ID sans nom. Mais d'autres distributions traitent cet ID différemment, donc les programmes portables ne devraient pas dépendre de cette assignation.

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

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

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

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

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

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

Le fichier `/var/log/wtmp` enregistre toutes les connexions et les déconnexions. Le fichier `/var/log/lastlog` enregistre le moment où chaque utilisateur s'est connecté pour la dernière fois. Le fichier `/var/log/faillog` enregistre les échecs de connexion. Le fichier `/var/log/btmp` enregistre les mauvaises tentatives de connexion.

Note

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

Note

Les fichiers `utmp`, `wtmp`, `btmp` et `lastlog` utilisent des entiers sur 32 bits pour l'horodatage et ils seront complètement cassés après l'année 2038. De nombreux paquets ont arrêté de les utiliser et d'autres paquets vont arrêter de les utiliser. Il vaut mieux les considérer comme obsolètes.

7.7. Gettext-0.26

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

Temps de construction approximatif: 1,5 SBU
Espace disque requis: 463 Mo

7.7.1. Installation de Gettext

Pour notre ensemble temporaire d'outils, nous avons besoin uniquement d'installer trois programmes de Gettext.

Préparez la compilation de Gettext :

```
./configure --disable-shared
```

Voici la signification de l'option de configuration :

`--disable-shared`
Nous n'avons pas besoin d'installer les bibliothèques partagées de Gettext pour l'instant, donc il n'y a pas besoin de les construire.

Compilez le paquet :

```
make
```

Installez les programmes **msgfmt**, **msgmerge** et **xgettext** :

```
cp -v gettext-tools/src/{msgfmt,msgmerge,xgettext} /usr/bin
```

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

7.8. Bison-3.8.2

Le paquet Bison contient un générateur d'analyseurs.

Temps de construction approximatif: 0,2 SBU
Espace disque requis: 58 Mo

7.8.1. Installation de Bison

Préparez la compilation de Bison :

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

Voici la signification de la nouvelle option de configuration :

`--docdir=/usr/share/doc/bison-3.8.2`
Cela dit au système de construction d'installer la documentation de bison dans un répertoire versionné.

Compilez le paquet :

```
make
```

Installez le paquet :

```
make install
```

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

7.9. Perl-5.42.0

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

Temps de construction approximatif: 0,6 SBU
Espace disque requis: 295 Mo

7.9.1. Installation de Perl

Préparez la compilation de Perl :

```
sh Configure -des                                         \
             -D prefix=/usr                               \
             -D vendorprefix=/usr                         \
             -D useshrplib                                \
             -D privlib=/usr/lib/perl5/5.42/core_perl     \
             -D archlib=/usr/lib/perl5/5.42/core_perl     \
             -D sitelib=/usr/lib/perl5/5.42/site_perl     \
             -D sitearch=/usr/lib/perl5/5.42/site_perl    \
             -D vendorlib=/usr/lib/perl5/5.42/vendor_perl \
             -D vendorarch=/usr/lib/perl5/5.42/vendor_perl
```

Voici la signification des options de configuration :

`-des`
C'est la combinaison de trois options : -d utilise les valeurs par défaut pour tous les éléments ; -e s'assure que toutes les tâches sont effectuées ; -s rend silencieuses les sorties non importantes.

`-D vendorprefix=/usr`
Ceci s'assure que **perl** sait comment dire aux paquets où ils devraient installer leurs modules Perl.

`-D useshrplib`
Construit la `libperl` requise par certains modules Perl en tant que bibliothèque partagée, au lieu d'une bibliothèque statique.

`-D privlib`, *`-D archlib`*, *`-D sitelib`*, *`...`*
Ces paramètres définissent où Perl cherche les modules installés. Les auteurs de LFS ont choisi de les mettre dans une structure de répertoire basée sur la version MAJEURE.MINEURE de Perl (5.42), ce qui permet de mettre à jour Perl vers de nouvelles versions de correctif (le niveau de correctif est la dernière partie séparée par un point dans la chaine de version complète comme 5.42.0) sans réinstaller tous les modules.

Compilez le paquet :

```
make
```

Installez le paquet :

```
make install
```

Les détails sur ce paquet sont disponibles dans Section 8.43.2, « Contenu de Perl. »

7.10. Python-3.13.7

Le paquet Python 3 contient l'environnement de développement Python. Il est utile pour la programmation orientée objet, écrire des scripts, prototyper de plus grands programmes ou pour développer des applications complètes.

Temps de construction approximatif: 0,5 SBU
Espace disque requis: 546 Mo

7.10.1. Installation de Python

Note

Il y a deux fichiers de paquet dont le nom commence par le préfixe « python ». Celui à extraire est `Python-3.13.7.tar.xz` (attention à la majuscule sur la première lettre).

Préparez la compilation de Python :

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

Voici la signification de l'option de configuration :

`--enable-shared`
Ce paramètre évite l'installation des bibliothèques statiques.

`--without-ensurepip`
Ce paramètre désactive l'installateur de paquets Python, qui n'est pas requise pour le moment.

`--without-static-libpython`
Ce paramètre évite la construction d'une grosse bibliothèque statique inutile.

Compilez le paquet :

```
make
```

Note

Certains modules Python 3 ne peuvent pas être construits à cause de dépendances qui ne sont pas encore installées. Pour le module `ssl`, un message `Python requires a OpenSSL 1.1.1 or newer` est affiché. Vous devriez ignorer ce message. Assurez-vous seulement que la commande **make** de plus haut niveau n'a pas échoué. Les modules facultatifs ne sont pas encore requis et ils seront construits dans le Chapitre 8.

Installez le paquet :

```
make install
```

Les détails sur ce paquet sont disponibles dans Section 8.51.2, « Contenu de Python 3. »

7.11. Texinfo-7.2

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

Temps de construction approximatif: 0,2 SBU
Espace disque requis: 152 Mo

7.11.1. Installation de Texinfo

Préparez la compilation de Texinfo :

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

Compilez le paquet :

```
make
```

Installez le paquet :

```
make install
```

Les détails sur ce paquet sont disponibles dans Section 8.72.2, « Contenu de Texinfo. »

7.12. Util-linux-2.41.1

Le paquet Util-linux contient divers programmes utilitaires.

Temps de construction approximatif: 0,2 SBU
Espace disque requis: 192 Mo

7.12.1. Installation d'Util-linux

Les FHS recommandent d'utiliser le répertoire `/var/lib/hwclock` au lieu du répertoire habituel `/etc` comme emplacement du fichier `adjtime`. Créez ce répertoire avec :

```
mkdir -pv /var/lib/hwclock
```

Préparez la compilation d'Util-linux :

```
./configure --libdir=/usr/lib     \
            --runstatedir=/run    \
            --disable-chfn-chsh   \
            --disable-login       \
            --disable-nologin     \
            --disable-su          \
            --disable-setpriv     \
            --disable-runuser     \
            --disable-pylibmount  \
            --disable-static      \
            --disable-liblastlog2 \
            --without-python      \
            ADJTIME_PATH=/var/lib/hwclock/adjtime \
            --docdir=/usr/share/doc/util-linux-2.41.1
```

Voici la signification des options de configuration :

`ADJTIME_PATH=/var/lib/hwclock/adjtime`
Cela configure l'emplacement du fichier enregistrant les informations sur l'horloge matérielle en accord avec la FHS. Cela n'est pas strictement requis pour cet outil temporaire, mais cela évite de créer un fichier ailleurs qui ne sera pas remplacé ou supprimé en construisant le paquet util-linux final.

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

`--disable-*`
Ces paramètres évitent des avertissements à propos de la construction des composants qui requièrent des paquets qui ne sont pas dans LFS ou pas encore installés.

`--without-python`
Ce paramètre désactive l'utilisation de Python. Cela évite de construire des liaisons inutiles.

`runstatedir=/run`
Ce paramètre indique l'emplacement du socket utilisé par **uuidd** et `libuuid`.

Compilez le paquet :

```
make
```

Installez le paquet :

```
make install
```

Les détails sur ce paquet sont disponibles dans Section 8.79.2, « Contenu d'Util-linux. »

7.13. Nettoyage et Sauvegarde du système temporaire

7.13.1. Nettoyage

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

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

Ensuite, sur un système Linux moderne, les fichiers .la de libtool ne sont utiles que pour libltdl. Aucune bibliothèque dans LFS ne sera chargée par libltdl, et certains fichiers .la sont connus pour causer des échecs à la construction de certains paquets de BLFS. Supprimez maintenant ces fichiers :

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

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

```
rm -rf /tools
```

7.13.2. Sauvegarde

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

Note

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

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

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

```
exit
```

Important

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

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

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

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

```
mountpoint -q $LFS/dev/shm && umount $LFS/dev/shm
umount $LFS/dev/pts
umount $LFS/{sys,proc,run,dev}
```

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

Remarquez que les instructions ci-dessous spécifient le répertoire personnel de l'utilisateur `root` sur le système hôte, qui se trouve généralement sur le système de fichiers racine. Remplacez `$HOME` par un répertoire de votre choix si vous ne voulez pas stocker la sauvegarde dans le répertoire personnel de `root`.

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

Note

Comme l'archive de sauvegarde est compressée, elle prend un temps relativement long (plus de 10 minutes) même sur un système raisonnablement rapide.

```
cd $LFS
tar -cJpf $HOME/lfs-temp-tools-12.4.tar.xz .
```

Note

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

7.13.3. Restauration du système

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

Avertissement

Les commandes suivantes sont extrêmement dangereuses. Si vous lancez **rm -rf ./*** en tant que `root` et que vous ne vous êtes pas déplacés dans le répertoire $LFS ou que la variable d'environnement `LFS` n'est pas définie pour l'utilisateur `root`, elle détruira votre système complet. ON VOUS AURA PRÉVENU.

```
cd $LFS
rm -rf ./*
tar -xpf $HOME/lfs-temp-tools-12.4.tar.xz
```

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

Important

Si vous quittez l'environnement chroot pour créer une sauvegarde ou si vous recommencez à construire à partir d'une sauvegarde, rappelez-vous de vérifier que les systèmes de fichiers virtuels sont toujours montés (**findmnt | grep $LFS**). S'ils ne sont pas montés, remontez-les maintenant comme décrit dans le Section 7.3, « Préparer les systèmes de fichiers virtuels du noyau » et entrez de nouveau dans l'environnement chroot (voir le Section 7.4, « Entrer dans l'environnement chroot ») avant de continuer.

Partie IV. Construction du système LFS

Chapitre 8. Installer les logiciels du système de base

8.1. Introduction

Dans ce chapitre, nous commençons la construction du système LFS pour de bon.

Nous arrivons à la dernière étape de l'installation de ce logiciel. Bien que, dans beaucoup de cas, les instructions d'installation pourraient être plus courtes et plus génériques, nous avons opté pour fournir les instructions complètes pour chaque paquet et minimiser ainsi les possibilités d'erreurs. La clé pour apprendre ce qui fait fonctionner un système Linux est de savoir à quoi sert chaque paquet et pourquoi vous (ou le système) en avez besoin.

L'utilisation d'optimisations personnalisées est déconseillée. Bien qu'elles puissent accélérer légèrement l'exécution des programmes, elles peuvent aussi poser des problèmes lors de leur compilation ou leur exécution. Si un paquet refuse de se compiler lors de l'utilisation d'optimisations, essayez de le compiler sans optimisation pour voir si cela corrige le problème. Même si le paquet parvient à se compiler avec les optimisations, il risque de mal se compiler à cause des interactions complexes entre le code et les outils de construction. Remarquez aussi que l'utilisation des options `-march` et `-mtune` avec des valeurs non indiquées dans LFS n'a pas été testée. Cela peut entrainer des problèmes avec les paquets de la chaîne d'outils (Binutils, GCC et Glibc). Le petit potentiel de gains obtenu en utilisant les optimisations de compilation est souvent minime comparé aux risques. Les utilisateurs construisant un système LFS pour la première fois sont encouragés à construire sans optimisations personnalisées.

Cependant, il convient de garder les optimisations activées par la configuration par défaut. De plus, LFS active parfois des configurations optimisées fournies par un paquet mais qui ne sont pas activées par défaut. Les mainteneurs de paquets ont déjà testé ces configurations et les ont jugées sans danger, il y a donc peu de chances qu'elles cassent la construction. En général, la configuration par défaut active déjà les options `-O2` ou `-O3`, le système ainsi obtenu pourra toujours fonctionner rapidement sans optimisation personnalisée et sera stable.

Avant les instructions d'installation, chaque page d'installation fournit des informations sur le paquet, incluant une description concise de ce qu'il contient, approximativement combien de temps prendra la construction et combien d'espace disque est nécessaire pendant le processus de construction. Après les instructions d'installation, il y a une liste de programmes et de bibliothèques (avec quelques brèves descriptions de ceux-ci) que le paquet installe.

Note

Les valeurs SBU et l'espace disque requis incluent les données de suites de tests pour tous les paquets de Chapitre 8 auxquels elles sont applicables. Les valeurs de SBU ont été calculées avec quatre cœurs CPU (-j4) pour toutes les opérations, sauf mention contraire.

8.1.1. À propos des bibliothèques

En général, les éditeurs de LFS déconseillent la construction et l'installation de bibliothèques statiques. La plupart des bibliothèques statiques ont été rendues obsolètes dans les systèmes Linux modernes. Par ailleurs la liaison statique de bibliothèques dans un programme peut être nuisible. Si une mise à jour des bibliothèques est nécessaire pour résoudre un problème de sécurité, tous les programmes qui utilisent cette bibliothèque vont devoir être liés à nouveau à la nouvelle bibliothèque. Comme l'utilisation de bibliothèques statiques n'est pas toujours évidente, on ne connaît même pas forcément les programmes adéquats (et les procédures requises pour faire la liaison).

Dans les procédures de ce chapitre, nous retirons ou désactivons l'installation de la plupart des bibliothèques statiques. Généralement cela ce fait en activant l'option `--disable-static` lors de l'exécution de **configure**. Dans certains cas, d'autres moyens sont nécessaires. Dans de rares cas, l'utilisation de bibliothèques statiques reste essentielle pour le processus de construction de paquets, surtout pour glibc et gcc.

Pour obtenir plus d'informations à propos des bibliothèques, regardez la discussion *Bibliothèques : statiques ou partagées ?* dans le livre BLFS.

8.2. Gestion des paquets

On nous a souvent demandé d'ajouter la gestion des paquets au livre LFS. Un gestionnaire de paquets permet de suivre l'installation des fichiers, simplifiant ainsi la suppression ou la mise à jour des paquets. Un bon gestionnaire de paquets gère également les fichiers de configuration pour conserver la configuration spécifique de l'utilisateur lorsque le paquet est réinstallé ou mis à jour. Avant toute chose, cette section ne parle pas et ne recommande pas un gestionnaire de paquets en particulier. Elle résume les techniques les plus populaires et leur fonctionnement. Le gestionnaire de paquets qui vous convient le mieux peut se trouver parmi ces techniques ou être une combinaison d'une ou plusieurs techniques. Cette section mentionne brièvement les problèmes qui peuvent survenir lors de la mise à jour de paquets.

Voici la raison pour laquelle aucun gestionnaire de paquets n'est mentionné dans LFS ou BLFS :

- La gestion des paquets ne fait pas partie de l'objectif de ces livres, qui visent à apprendre aux utilisateurs à construire un système Linux.
- Il existe de nombreuses solutions pour gérer des paquets. Chacune a ses points forts et ses points faibles. En trouver une qui satisfait tout le monde est difficile.

Nous avons évoqué quelques astuces sur le sujet de la gestion des paquets. Consultez le *Hints Project* et trouvez le gestionnaire qui correspond à vos besoins.

8.2.1. Problèmes de mise à jour

Un gestionnaire de paquets facilite la mise à jour des nouvelles versions au moment de leur sortie. Généralement, les instructions contenues dans les livres LFS et BLFS peuvent être utilisées pour mettre à jour les paquets vers de nouvelles versions. Voici quelques points à connaître pour mettre à jour vos paquets, spécifiquement sur un système en cours de fonctionnement.

- Si le noyau Linux doit être mis à jour, et passer par exemple de la version 5.10.17 à la version 5.10.18 ou 5.11.1, vous n'avez pas besoin de reconstruire d'autres éléments. Le système continuera de fonctionner correctement grâce à l'interface bien définie entre le noyau et l'espace utilisateur. Plus précisément, les en-têtes de l'API Linux n'ont pas besoin d'être mis à jour en même temps que le noyau. Vous devrez simplement redémarrer votre système pour utiliser le noyau à jour.
- Si Glibc doit être mis à jour vers une nouvelle version (p. ex. de Glibc-2.36 à Glibc-2.42), des étapes supplémentaires sont requises pour éviter de casser le système. Consultez Section 8.5, « Glibc-2.42 » pour plus de détails.
- Si un paquet contenant une bibliothèque partagée est mis à jour et si le nom de cette dernière[1] est modifié, alors les paquets liés dynamiquement à la bibliothèque devront être recompilés pour être liés à la nouvelle bibliothèque. Par exemple, prenez un paquet foo-1.2.3 qui installe une bibliothèque partagée appelée `libfoo.so.1`. Partons du principe que vous mettez à jour le paquet avec une nouvelle version foo-1.2.4 qui installe une bibliothèque partagée appelée `libfoo.so.2`. Dans ce cas, tous les paquets liés dynamiquement à `libfoo.so.1` doivent être recompilés pour être liés à `libfoo.so.2`. Vous ne devez pas supprimer les anciennes bibliothèques avant que les paquets indépendants ne soient tous recompilés.
- Si un paquet est (directement ou indirectement) lié à la fois à l'ancien et au nouveau nom d'une bibliothèque partagée (par exemple, le paquet se lie à la fois à `libfoo.so.2` et à `libbar.so.1`, alors que cette dernière est un lien vers `libfoo.so.3`), le paquet peut ne pas fonctionner correctement car les différentes versions des bibliothèques partagées fournissent des définitions incompatibles pour certains noms de symboles. Cela peut arriver en recompilant certains paquets, mais en en oubliant d'autres, liés à d'anciennes bibliothèques après la

---

Le nom d'une bibliothèque partagée est la chaîne codée dans l'entrée `DT_SONAME` de la section dynamique de son fichier ELF. Vous pouvez le récupérer avec la commande **readelf -d *<fichier de bibliothèque>* | grep SONAME**. Dans la plupart des cas, il a pour suffixe `.so.<numéro de version>`, mais il y a des cas où il contient plusieurs numéros de version (comme `libbz2.so.1.0`), où il contient le numéro de version avant le suffixe `.so` (comme `libbfd-2.45`) ou ne contient aucun numéro de version (par exemple `libmemusage.so`). En général, il n'y a pas de lien entre la version du paquet et le numéro de version dans le nom de la bibliothèque.

mise à jour du paquet qui fournit la bibliothèque partagée. Pour éviter ce problème, vous devrez recompiler tous les paquets liés à une bibliothèque partagée qui possède une nouvelle révision (p. ex. libfoo.so.2 devient libfoo.so.3) le plus vite possible.

- Si vous mettez à jour un paquet qui contient une bibliothèque partagée et que le nom de la bibliothèque ne change pas, mais que le numéro de version du **fichier** de la bibliothèque décroît (par exemple le nom reste `libfoo.so.1`, mais le nom du fichier de la bibliothèque change de `libfoo.so.1.25` à `libfoo.so.1.24`), vous devez supprimer le fichier de bibliothèque de la version précédente (`libfoo.so.1.25` dans ce cas). Sinon, en exécutant **ldconfig** manuellement via la ligne de commande, ou en installant un paquet, vous réinitialiserez le lien symbolique `libfoo.so.1` vers l'ancien fichier de bibliothèque, car sa version est « plus récente », puisque le numéro est plus grand. Cette situation arrive quand vous installez une version précédente d'un paquet, ou que l'auteur du paquet change de pratique de nommage des versions.
- Si vous mettez à jour un paquet qui contient une bibliothèque partagée et que le nom de la bibliothèque ne change pas, mais qu'un problème important, comme une vulnérabilité de sécurité, est corrigé, tous les programmes en cours d'exécution liés à la bibliothèque partagée doivent être redémarrés. La commande suivante, lancée en tant qu'utilisateur `root` après la mise à jour, affiche les processus qui utilisent les anciennes versions de ces bibliothèques (remplacez *`libfoo`* par le nom de la bibliothèque) :

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

  Si OpenSSH est utilisé pour accéder au système et qu'il est lié à la bibliothèque mise à jour, vous devez redémarrer le service **sshd**, vous déconnecter, vous reconnecter et relancer la commande précédente pour confirmer qu'aucun processus n'utilise les bibliothèques supprimées.
- Si un programme ou une bibliothèque partagée est écrasé, les processus utilisant le code ou les données du programme ou de la bibliothèque peuvent planter. La bonne manière de mettre à jour un programme ou une bibliothèque partagée sans interruption anormale du processus est de le supprimer d'abord, puis d'installer la nouvelle version. La commande **install** fournie par coreutils implémente déjà cela et la plupart des paquets l'utilisent pour installer des binaires et des bibliothèques. Cela signifie que vous n'aurez pas ce problème la plupart du temps. Cependant, le processus d'installation de certains paquets, notamment SpiderMonkey dans BLFS, se contente de réécrire sur le fichier s'il existe déjà et cause un crash, donc il est plus prudent de sauvegarder votre travail et de fermer les processus inutiles avant de mettre à jour un paquet.

8.2.2. Techniques de gestion des paquets

Voici une liste des techniques les plus courantes en gestion de paquets. Avant de choisir un gestionnaire de paquets, cherchez les différentes techniques et notamment les points faibles de chaque système.

8.2.2.1. Tout est dans ma tête !

Oui, c'est une technique de gestion des paquets. Certains n'ont pas besoin d'un gestionnaire de paquets parce qu'ils connaissent très bien les paquets et connaissent les fichiers installés pour chaque paquet. D'autres n'en ont pas besoin parce qu'ils planifient la reconstruction entière de LFS dès qu'un paquet est modifié.

8.2.2.2. Installation dans des répertoires distincts

C'est une technique de gestion des paquets simple qui ne nécessite aucun paquet supplémentaire pour gérer les installations. Chaque paquet est installé dans un répertoire distinct. Par exemple, le paquet foo-1.1 est installé dans `/opt/foo-1.1` et un lien symbolique est créé depuis `/opt/foo` vers `/opt/foo-1.1`. Lors de la mise à jour vers la nouvelle version foo-1.2, elle est installée dans `/opt/foo-1.2` et l'ancien lien symbolique est remplacé par le lien symbolique qui mène à la nouvelle version.

Les variables d'environnement telles que `PATH`, `MANPATH`, `INFOPATH`, `PKG_CONFIG_PATH`, `CPPFLAGS`, `LDFLAGS` et le fichier de configuration `/etc/ld.so.conf` pourraient avoir besoin d'être étendus pour inclure les sous-répertoire correspondants dans `opt/foo-x.y`.

Ce système est utilisé par le livre BLFS pour installer certains très gros paquets pour faciliter leur mise à jour. Si vous installez plus de quelques paquets, ce système devient ingérable. En plus, certains paquets (par exemple les en-têtes de l'API Linux et Glibc) peuvent ne pas fonctionner correctement avec ce système. **N'utilisez jamais ce système pour le système entier**.

8.2.2.3. Gestion de paquets par lien symbolique

Il s'agit d'une variante de la technique précédente. Chaque paquet est installé de façon similaire au système précédent. Mais au lieu de créer le lien symbolique avec un nom générique pour chaque paquet, chaque fichier dispose d'un lien symbolique dans la hiérarchie `/usr`. Il n'y a alors plus besoin d'étendre les variables d'environnement. Même si les liens symboliques peuvent être créés par l'utilisateur, beaucoup de gestionnaires de paquets utilisent cette approche pour automatiser la création de liens symboliques. Parmi les plus populaires, on retrouve Stow, Epkg, Graft et Depot.

Le script d'installation doit être faussé de façon à ce que chaque paquet pense qu'il est installé dans le répertoire `/usr`, alors qu'en réalité il est installé dans l'arborescence `/usr/pkg`. Réaliser l'installation de cette manière n'est généralement pas tâche aisée. Par exemple, supposons que vous installez un paquet libfoo-1.1. Les instructions suivantes peuvent ne pas installer correctement le paquet :

```
./configure --prefix=/usr/pkg/libfoo/1.1
make
make install
```

L'installation fonctionnera, mais les paquets dépendants peuvent ne pas se lier à libfoo de la manière prévue. Si vous compilez un paquet lié à libfoo, vous verrez que qu'il est aussi lié à `/usr/pkg/libfoo/1.1/lib/libfoo.so.1` au lieu de `/usr/lib/libfoo.so.1` comme attendu. La bonne approche consiste à utiliser la stratégie `DESTDIR` pour diriger l'installation du paquet. Cette approche fonctionne ainsi :

```
./configure --prefix=/usr
make
make DESTDIR=/usr/pkg/libfoo/1.1 install
```

La plupart des paquets prennent en charge cette approche, mais elle pose problème à certains utilisateurs. Pour les paquets non compatibles, vous pouvez soit les installer manuellement, soit opter pour une méthode plus simple en installant les paquets posant problème dans `/opt`.

8.2.2.4. Basé sur l'horodatage

Avec cette technique, un fichier est horodaté avant l'installation du paquet. Après l'installation, une simple exécution de la commande **find** avec les options appropriées peut générer une trace de tous les fichiers installés après la création du fichier horodaté. Le gestionnaire de paquets install-log utilise ce système.

Bien que ce schéma ait l'avantage d'être simple, il a deux inconvénients. Si à l'installation les fichiers sont installés dans autre horodatage que celui de l'heure actuelle, ils ne seront pas suivis par le gestionnaire de paquets. De plus, ce système peut être utilisé seulement lorsqu'un seul paquet est installé à la fois. Les traces ne sont pas fiables si deux paquets sont installés depuis deux consoles différentes.

8.2.2.5. Tracer les scripts d'installation

Avec cette approche, les commandes que les scripts d'installation exécutent sont enregistrées. Il existe deux techniques :

Vous pouvez initialiser la variable d'environnement `LD_PRELOAD` pour qu'elle pointe vers une bibliothèque à précharger avant l'installation. Lors de l'installation de cette dernière, la bibliothèque trace les paquets en cours d'installation en s'attachant aux différents exécutables comme **cp**, **install**, **mv** et trace les appels système qui modifient le système de fichiers. Pour que cette approche fonctionne, tous les exécutables ont besoin d'être liés

dynamiquement sans bit suid ou sgid. Le préchargement de la bibliothèque peut provoquer des effets secondaires indésirables lors de l'installation ; effectuez donc quelques tests pour vous assurer que le gestionnaire de paquets n’endommage rien et trace bien les fichiers nécessaires.

La seconde technique consiste à utiliser la commande **strace**, qui trace tous les appels du système effectués pendant l'exécution des scripts d'installation.

8.2.2.6. Créer des archives de paquets

Dans ce système, l'installation d'un paquet est déplacé pour de faux dans un répertoire séparé comme décrit plus haut. Après l'installation, une archive du paquet est créée grâce aux fichiers installés. L'archive est ensuite utilisée pour installer le paquet, soit sur la machine locale, soit sur d'autres machines.

Cette approche est utilisée par la plupart des gestionnaires de paquets trouvés dans les distributions commerciales. Les exemples de gestionnaires qui suivent cette approche sont RPM, qui d'ailleurs est requis par la *Spécification de base de Linux Standard*, pkg-utils, apt de Debian et le système de portage de Gentoo. Une astuce montrant comment adopter ce style de gestion de paquets pour les systèmes LFS se trouve ici : *http://www.fr.linuxfromscratch.org/view/astuces/fakeroot-fr.txt*.

La création de fichiers de paquet qui incluent des informations de dépendance est complexe et va au-delà de l'objectif de LFS.

Slackware utilise un système basé sur **tar** pour les archives de paquets. Ce système ne gère volontairement pas les dépendances de paquets car d'autres gestionnaires de paquets plus complexes le font. Pour plus d’informations sur la gestion des paquets, voir *https://www.slackbook.org/html/package-management.html*.

8.2.2.7. Gestion basée sur les utilisateurs

Cette méthode, unique à LFS, a été décrite par Matthias Benkmann et est disponible sur le *Hints Project*. Selon cette méthode, chaque paquet est installé en tant qu'utilisateur séparé dans les emplacements standards. Les fichiers appartenant à un paquet sont facilement identifiables grâce à l'identifiant de l'utilisateur. Les avantages et inconvénients de cette approche sont trop complexes pour pouvoir tous les décrire dans cette section. Pour plus d’informations, voir l'astuce sur *http://www.fr.linuxfromscratch.org/view/astuces/gestionnaire-paquets-utilisateur.txt*.

8.2.3. Déployer LFS sur plusieurs systèmes

Le fait qu’il n'y ait pas de fichiers dépendants de la position des fichiers sur un système de disque est l’un des avantages du système LFS. Cloner la construction d'un système LFS sur un autre ordinateur avec une architecture similaire au système de base se résume à utiliser la commande **tar** sur la partition LFS qui contient le répertoire racine (environ 900 Mo décompressés pour une construction LFS de base), en copiant ce fichier via un transfert par réseau ou par CD-ROM ou clé USB vers le nouveau système et en le décompressant. Vous devez ensuite modifier quelques fichiers de configuration. Les fichiers de configuration qui nécessitent une mise à jour comprennent : `/etc/hosts`, `/etc/fstab`, `/etc/passwd`, `/etc/group`, `/etc/shadow`, `/etc/ld.so.conf`, `/etc/sysconfig/rc.site`, `/etc/sysconfig/network`, et `/etc/sysconfig/ifconfig.eth0`.

Vous pouvez construire un noyau personnalisé pour le nouveau système selon les différences dans le système matériel et la configuration du noyau initial.

Important

Si vous voulez déployer le système LFS sur un système avec un autre CPU, vous devez suivre les notes sur la modification de l'optimisation dépendant de l'architecture lors de la construction de Section 8.21, « GMP-6.3.0 » et Section 8.50, « Libffi-3.5.2 » pour produire des bibliothèques compatibles à la fois avec le système hôte et les systèmes sur lesquels vous allez déployer le système LFS. Dans le cas contraire, vous obtiendrez des erreurs de type `Illegal Instruction` sur le système LFS.

Enfin, vous devez rendre le nouveau système démarrable via Section 10.4, « Utiliser GRUB pour paramétrer le processus de démarrage ».

8.3. Man-pages-6.15

Le paquet Man-pages contient environ 2 400 pages de manuel.

Temps de construction approximatif: 0,1 SBU
Espace disque requis: 52 Mo

8.3.1. Installation de Man-pages

Supprimez deux pages de manuel pour les fonctions de hashage des mots de passe. Libxcrypt fournira une meilleure version de ces pages de manuel :

```
rm -v man3/crypt*
```

Installez Man-pages en lançant :

```
make -R GIT=false prefix=/usr install
```

Voici la signification des options :

`-R`
Cela évite que **make** ne mette en place des variables prédéfinies. Le système de construction des pages de manuel ne fonctionne pas bien avec ces variables, mais actuellement il n'y a pas de moyen de les désactiver à part passer `-R` explicitement sur la ligne de commande.

`GIT=false`
Cela évite que le système de construction n'affiche de nombreuses lignes de `git : commande introuvable`.

8.3.2. Contenu de Man-pages

Fichiers installés: différentes pages de manuel

Descriptions courtes

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

8.4. Iana-Etc-20250807

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

Temps de construction approximatif: moins de 0,1 SBU
Espace disque requis: 4,8 Mo

8.4.1. Installation de Iana-Etc

Pour ce paquet, nous avons uniquement besoin de copier les fichiers à leur place :

```
cp services protocols /etc
```

8.4.2. Contenu de Iana-Etc

Fichiers installés: /etc/protocols et /etc/services

Descriptions courtes

`/etc/protocols` Décrit les différents protocoles Internet DARPA disponibles à partir du sous-système TCP/IP

`/etc/services` Fournit une correspondance entre des noms de services internet et leurs numéros de ports et types de protocoles affectés

8.5. Glibc-2.42

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

Temps de construction approximatif: 12 SBU
Espace disque requis: 3,3 Go

8.5.1. Installation de Glibc

Certains programmes de Glibc utilisent le répertoire non conforme au FHS `/var/db` pour stocker leurs données d'exécution. Appliquez le correctif suivant pour que ces programmes stockent leurs données à des emplacements conformes au FHS :

```
patch -Np1 -i ../glibc-2.42-fhs-1.patch
```

Corrigez maintenant un problème qui peut casser Valgrind dans BLFS :

```
sed -e '/unistd.h/i #include <string.h>' \
    -e '/libc_rwlock_init/c\
  __libc_rwlock_define_initialized (, reset_lock);\
  memcpy (&lock, &reset_lock, sizeof (lock));' \
    -i stdlib/abort.c
```

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

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

Assurez-vous que les utilitaires **ldconfig** et **sln** s'installent dans le répertoire `/usr/sbin` :

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

Préparez la compilation de Glibc :

```
../configure --prefix=/usr                      \
             --disable-werror                   \
             --disable-nscd                     \
             libc_cv_slibdir=/usr/lib           \
             --enable-stack-protector=strong \
             --enable-kernel=5.4
```

Voici la signification des options de configuration :

`--disable-werror`
Désactive l'option -Werror passée à GCC. Ceci est nécessaire pour lancer la suite de tests.

`--enable-kernel=5.4`
Indique au système de construction que cette Glibc peut être utilisée avec les noyaux aussi vieux que 5.4 (maximum). Cela permet de générer des contournements au cas où on ne peut pas utiliser un appel système introduit dans une version ultérieure.

`--enable-stack-protector=strong`
Cette option améliore la sécurité du système en ajoutant du code supplémentaire pour vérifier les dépassements de tampon, comme dans les attaques par dépassement de pile. Remarquez que Glibc remplace toujours la valeur par défaut de GCC, donc cette option est nécessaire, même si nous avons déjà spécifié l'option `--enable-default-ssp` pour GCC.

--disable-nscd

Ne pas construire le démon de cache de service de nom qui n'est plus utilisé.

libc_cv_slibdir=/usr/lib

Indique la bibliothèque adaptée pour chaque système. Nous ne voulons pas utiliser lib64.

Compilez le paquet :

```
make
```

Important

Dans cette section, la suite de tests de Glibc est considérée comme critique. Ne l'ignorez sous aucun prétexte.

En général, quelques tests ne réussissent pas, mais vous pouvez le plus souvent ignorer les erreurs listées ci-dessous.

```
make check
```

Vous verrez probablement quelques erreurs lors des tests. La suite de tests de Glibc est quelque peu dépendante du système hôte. Vous pouvez généralement ignorer quelques erreurs parmi les plus de 6 000 tests. Voici une liste des problèmes les plus fréquents dans les versions récentes de LFS :

- *io/tst-lchmod* est connu pour échouer dans l'environnement chroot de LFS.
- *misc/tst-preadvwritev2* et *misc/tst-preadvwritev64v2* sont connus pour échouer si le noyau hôte est Linux-6.14 ou supérieur.
- Certains tests, par exemple *nss/tst-nss-files-hosts-multi* et *nptl/tst-thread-affinity** sont connus pour échouer à cause d'un délai d'attente dépassé (surtout quand le système est assez lent ou exécute la suite de tests avec plusieurs tâches make en parallèle). Ces tests peuvent être identifiés avec :

  ```
  grep "Timed out" $(find -name \*.out)
  ```

  Il est possible de relancer un test unique avec un plus grand délai d'attente avec **TIMEOUTFACTOR=<*factor*> make test t=<*nom du test*>**. Par exemple, **TIMEOUTFACTOR=10 make test t=nss/tst-nss-files-hosts-multi** relancera *nss/tst-nss-files-hosts-multi* avec dix fois plus de temps que l'original.
- De plus, certains tests peuvent échouer avec un modèle de CPU assez ancien (par exemple *elf/tst-cpu-features-cpuinfo*) ou une version du noyau hôte assez ancienne (par exemple *stdlib/tst-arc4random-thread*).

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

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

Corrigez le Makefile généré pour éviter un test de cohérence obsolète qui échoue avec une configuration moderne de Glibc :

```
sed '/test-installation/s@$(PERL)@echo not running@' -i ../Makefile
```

Important

Si vous mettez à jour Glibc vers une nouvelle version mineure (par exemple, de Glibc-2.36 à Glibc-2.42) sur un système LFS qui tourne, vous devez prendre des précautions supplémentaires pour éviter de casser votre système :

- Mettre à jour Glibc sur un système LFS avant 11.0 (exclusif) n'est pas pris en charge. Reconstruisez LFS si vous avez un système LFS aussi vieux et que vous avez besoin d'une nouvelle Glibc.
- Si vous mettez à jour depuis un système LFS avant 12.0 (exclusif), installez Libxcrypt en suivant Section 8.27, « Libxcrypt-4.4.38. ». En plus d'une installation de Libxcrypt normale, **vous DEVEZ suivre la note dans la section Libxcrypt pour installer `libcrypt.so.1*` (ce qui remplace `libcrypt.so.1` de l'installation de Glibc précédente)**.
- Si vous mettez à jour depuis un système LFS avant 12.1 (exclusif), supprimez le programme **nscd** :

```
rm -f /usr/sbin/nscd
```

- Mettez à jour le noyau et redémarrez s'il est plus vieux que 5.4 (vérifiez la version actuelle avec **uname -r**) ou si vous voulez quand même le mettre à jour, en suivant Section 10.3, « Linux-6.16.1. »
- Mettez à jour les en-têtes de l'API du noyau s'il est plus vieux que 5.4 (vérifiez la version actuelle avec **cat /usr/include/linux/version.h**) ou si vous voulez mettre à jour quand même, en suivant Section 5.4, « Linux-6.16.1 API Headers » (mais en supprimant `$LFS` de la commande **cp**).
- Effectuez une installation `DESTDIR` et mettez à jour les bibliothèques Glibc partagées sur le système avec une seule commande **install** :

```
make DESTDIR=$PWD/dest install
install -vm755 dest/usr/lib/*.so.* /usr/lib
```

Il est impératif de suivre à la lettre ces étapes à moins de comprendre complètement ce que vous faites. **Toute déviation inattendue peut rendre le système complètement instable. VOUS ÊTES PRÉVENU.**

Continuez ensuite à exécutez la commande **make install**, la commande **sed** pour `/usr/bin/ldd` et les commandes pour installer les paramètres linguistiques. Une fois qu'elles ont fini, redémarrez le système immédiatement.

Une fois le système correctement redémarré, si vous lancez un système LFS datant d'avant la 12.0 (non compris) où GCC n'a pas été construit avec l'option *`--disable-fixincludes`*, déplacez deux en-têtes de GCC vers un meilleur emplacement et supprimez les copies « corrigées » obsolètes des en-têtes de Glibc :

```
DIR=$(dirname $(gcc -print-libgcc-file-name))
[ -e $DIR/include/limits.h ]    || mv $DIR/include{-fixed,}/limits.h
[ -e $DIR/include/syslimits.h ] || mv $DIR/include{-fixed,}/syslimits.h
rm -rfv $DIR/include-fixed/*
unset DIR
```

Installez le paquet :

```
make install
```

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

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

Ensuite, installez les locales qui permettent à votre système de répondre dans une langue différente. Aucune n'est indispensable, mais si certaines sont absentes, les suites de test des futurs paquets peuvent sauter des tests importants.

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

```
localedef -i C -f UTF-8 C.UTF-8
localedef -i cs_CZ -f UTF-8 cs_CZ.UTF-8
localedef -i de_DE -f ISO-8859-1 de_DE
localedef -i de_DE@euro -f ISO-8859-15 de_DE@euro
localedef -i de_DE -f UTF-8 de_DE.UTF-8
localedef -i el_GR -f ISO-8859-7 el_GR
localedef -i en_GB -f ISO-8859-1 en_GB
localedef -i en_GB -f UTF-8 en_GB.UTF-8
localedef -i en_HK -f ISO-8859-1 en_HK
localedef -i en_PH -f ISO-8859-1 en_PH
localedef -i en_US -f ISO-8859-1 en_US
localedef -i en_US -f UTF-8 en_US.UTF-8
localedef -i es_ES -f ISO-8859-15 es_ES@euro
localedef -i es_MX -f ISO-8859-1 es_MX
localedef -i fa_IR -f UTF-8 fa_IR
localedef -i fr_FR -f ISO-8859-1 fr_FR
localedef -i fr_FR@euro -f ISO-8859-15 fr_FR@euro
localedef -i fr_FR -f UTF-8 fr_FR.UTF-8
localedef -i is_IS -f ISO-8859-1 is_IS
localedef -i is_IS -f UTF-8 is_IS.UTF-8
localedef -i it_IT -f ISO-8859-1 it_IT
localedef -i it_IT -f ISO-8859-15 it_IT@euro
localedef -i it_IT -f UTF-8 it_IT.UTF-8
localedef -i ja_JP -f EUC-JP ja_JP
localedef -i ja_JP -f UTF-8 ja_JP.UTF-8
localedef -i nl_NL@euro -f ISO-8859-15 nl_NL@euro
localedef -i ru_RU -f KOI8-R ru_RU.KOI8-R
localedef -i ru_RU -f UTF-8 ru_RU.UTF-8
localedef -i se_NO -f UTF-8 se_NO.UTF-8
localedef -i ta_IN -f UTF-8 ta_IN.UTF-8
localedef -i tr_TR -f UTF-8 tr_TR.UTF-8
localedef -i zh_CN -f GB18030 zh_CN.GB18030
localedef -i zh_HK -f BIG5-HKSCS zh_HK.BIG5-HKSCS
localedef -i zh_TW -f UTF-8 zh_TW.UTF-8
```

En outre, installez la locale de votre pays, de votre langue et de votre encodage de caractères.

Sinon, vous pouvez installer les locales listées dans le fichier `glibc-2.42/localedata/SUPPORTED` (il inclut toutes les locales citées ci-dessus et bien plus) en une seule fois avec la commande suivante (qui prend un certain temps) :

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

Note

Glibc utilise maintenant libidn2 lors de la résolution de noms de domaines internationalisés. C'est une dépendance à l'exécution. Si cette fonctionnalité est requise, les instructions pour installer libidn2 se trouvent sur la *page libidn2 de BLFS*.

8.5.2. Configuration de Glibc

8.5.2.1. Ajout de nsswitch.conf

Le fichier `/etc/nsswitch.conf` doit être créé car les valeurs par défaut de Glibc ne fonctionnent pas correctement dans un environnement en réseau.

Créez un nouveau fichier `/etc/nsswitch.conf` en exécutant :

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

passwd: files
group: files
shadow: files

hosts: files dns
networks: files

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

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

8.5.2.2. Ajout des données de fuseaux horaires

Installez et configurez les données de fuseaux horaires en exécutant :

```
tar -xf ../../tzdata2025b.tar.gz

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

for tz in etcetera southamerica northamerica europe africa antarctica  \
          asia australasia backward; do
    zic -L /dev/null   -d $ZONEINFO       ${tz}
    zic -L /dev/null   -d $ZONEINFO/posix ${tz}
    zic -L leapseconds -d $ZONEINFO/right ${tz}
done

cp -v zone.tab zone1970.tab iso3166.tab $ZONEINFO
zic -d $ZONEINFO -p America/New_York
unset ZONEINFO tz
```

Voici la signification de la commande zic :

`zic -L /dev/null ...`
: Crée des fuseaux horaires posix sans secondes intercalaires. Par convention, on le met dans `zoneinfo` et dans `zoneinfo/posix`. Il faut indiquer les fuseaux horaires POSIX dans `zoneinfo`, sinon plusieurs suites de tests renverront des erreurs. Sur un système embarqué, où il y a peu de place et vous ne souhaitez pas mettre à jour les fuseaux horaires, vous pouvez économiser 1,9 Mo en n'utilisant pas le répertoire `posix`, mais certaines applications ou suites de tests pourraient échouer.

`zic -L leapseconds ...`
: Crée les fuseaux horaires adaptés incluant les secondes intercalaires. Sur un système embarqué, où il y a peu de place et vous ne souhaitez pas mettre à jour les fuseaux horaires, ou si vous ne trouvez pas important d'avoir la bonne heure, vous pouvez économiser 1,9 Mo en ne mettant pas de répertoire `right`.

`zic ... -p ...`
: Crée le fichier `posixrules`. Nous utilisons New York car POSIX exige que les règles de l'heure d'été respectent les règles américaines.

Lancez ce script pour déterminer dans quel fuseau horaire vous vous situez :

```
tzselect
```

Après quelques questions sur votre emplacement, le script affichera le nom du fuseau horaire (par exemple *Europe/Paris*). D'autres fuseaux horaires sont aussi listés dans le fichier `/usr/share/zoneinfo` comme *Canada/Eastern* ou *EST5EDT* qui ne sont pas identifiés par le script mais qui peuvent être utilisés.

Puis créez le fichier `/etc/localtime` en exécutant :

```
ln -sfv /usr/share/zoneinfo/<xxx> /etc/localtime
```

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

8.5.2.3. Configuration du chargeur dynamique

Par défaut, le chargeur dynamique (`/lib/ld-linux.so.2`) cherche dans `/usr/lib` les bibliothèques partagées nécessaires aux programmes lors de leur exécution. Néanmoins, s'il existe des bibliothèques dans d'autres répertoires que `/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 qui contiennent des bibliothèques supplémentaires, donc ajoutez-les au chemin de recherche du chargeur dynamique.

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

```
cat > /etc/ld.so.conf << "EOF"
# Begin /etc/ld.so.conf
/usr/local/lib
/opt/lib

EOF
```

Si vous le désirez, le chargeur dynamique peut également chercher un répertoire et inclure le contenu des fichiers qui s'y trouvent. Les fichiers de ce répertoire include sont en général constitués d'une ligne spécifiant le chemin vers la bibliothèque désirée. Pour ajouter cette possibilité, exécutez les commandes suivantes :

```
cat >> /etc/ld.so.conf << "EOF"
# Add an include directory
include /etc/ld.so.conf.d/*.conf

EOF
mkdir -pv /etc/ld.so.conf.d
```

8.5.3. Contenu de Glibc

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

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

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

Descriptions courtes

gencat Génère des catalogues de messages

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

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

iconv	Convertit l’encodage de caractères
iconvconfig	Crée des fichiers de configuration pour le module **iconv**
ldconfig	Configure les liens d’exécution de l’éditeur de liens dynamiques
ldd	Indique les bibliothèques partagées requises pour chaque programme ou bibliothèque partagée
lddlibc4	Assiste **ldd** avec des fichiers objets. Elle n'existe pas sur les architectures plus récentes comme x86_64
locale	Affiche diverses informations sur la locale actuelle
localedef	Compile les spécifications de la locale
makedb	Crée une base de données simple à partir d'une entrée textuelle
mtrace	Lit et interprète un fichier de trace et en affiche un résumé dans un format lisible par un humain
pcprofiledump	Affiche des informations générées par un profilage du PC
pldd	Liste les objets dynamiques partagés utilisés en exécutant des processus
sln	Un programme **ln** lié statiquement
sotruss	Retrace les procédures d'appel d'une bibliothèque partagée pour la commande spécifiée
sprof	Lit et affiche les données de profilage des objets partagés
tzselect	Demande à l'utilisateur l'emplacement géographique du système et donne la description du fuseau horaire correspondant
xtrace	Retrace 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`	Programme d'aide des bibliothèques partagées exécutables
`libBrokenLocale`	Utilisé en interne par Glibc comme une arme grossière pour résoudre les programmes cassés (comme certaines applications Motif). Voir les commentaires dans `glibc-2.42/locale/broken_cur_max.c` pour plus d'informations
`libanl`	Bibliothèque factice ne contenant aucune fonction. Il s’agissait d’une bibliothèque de recherche de noms asynchrone dont les fonctions se trouvent maintenant dans `libc`
`libc`	Bibliothèque du C principale
`libc_malloc_debug`	Active le test d'allocation de mémoire lorsqu'elle est préchargée
`libdl`	Bibliothèque factice ne contenant aucune fonction. Il s’agissait d’une bibliothèque d'interface de liaison dynamique dont les fonctions se trouvent maintenant dans `libc`
`libg`	Bibliothèque factice ne contenant aucune fonction. Il s’agissait d’une bibliothèque d'exécution pour **g++**
`libm`	Bibliothèque mathématique
`libmvec`	Bibliothèque de mathématiques vectorielles, liée si besoin quand `libm` est utilisée
`libmcheck`	Active le test d'allocation de mémoire lorsqu'on y relie quelque chose
`libmemusage`	Utilisé par **memusage** pour aider à la récupération d'informations sur l'utilisation de la mémoire par un programme
`libnsl`	Bibliothèque de services réseau, maintenant obsolète
`libnss_*`	Les modules du Name Service Switch, qui contient des fonctions de résolution de noms d'hôtes, de noms d'utilisateurs, de noms de groupes, d'alias, de services, de protocoles

et ainsi de suite. Chargée par la `libc` en fonction de la configuration présente dans le fichier `/etc/nsswitch.conf`

`libpcprofile` — Peut être préchargé pour profiler le PC d'un exécutable

`libpthread` — Bibliothèque factice ne contenant aucune fonction. Elle contenait les fonctions fournissant la plupart des interfaces spécifiées par les extensions POSIX.1c pour les fils d'exécution et des interfaces à sémaphores spécifiées par POSIX.1b pour le temps réel. Ces fonctions se trouvent maintenant dans `libc`

`libresolv` — Contient des fonctions de création, d'envoi et d'interprétation de paquets pour les serveurs de noms de domaine Internet

`librt` — Contient des fonctions fournissant la plupart des interfaces spécifiées par l'extension POSIX.1b pour le temps réel

`libthread_db` — Contient des fonctions utiles pour construire des débogueurs de programmes multi-threads

`libutil` — Bibliothèque factice ne contenant aucune fonction. Elle contenait du code pour les fonctions « standard » utilisées dans de nombreux utilitaires Unix différents. Ces fonctions se trouvent maintenant dans `libc`

8.6. Zlib-1.3.1

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

Temps de construction approximatif: moins de 0,1 SBU
Espace disque requis: 6,4 Mo

8.6.1. Installation de Zlib

Préparez la compilation de Zlib :

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

Compilez le paquet :

```
make
```

Pour tester les résultats, exécutez :

```
make check
```

Installez le paquet :

```
make install
```

Supprimez une bibliothèque statique inutile :

```
rm -fv /usr/lib/libz.a
```

8.6.2. Contenu de Zlib

Bibliothèques installées: libz.so

Descriptions courtes

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

8.7. Bzip2-1.0.8

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

Temps de construction approximatif: moins de 0,1 SBU
Espace disque requis: 7,3 Mo

8.7.1. Installation de Bzip2

Appliquez un correctif qui installera la documentation de ce paquet :

```
patch -Np1 -i ../bzip2-1.0.8-install_docs-1.patch
```

La commande suivante garantit l'installation de liens symboliques relatifs :

```
sed -i 's@\(ln -s -f \)$(PREFIX)/bin/@\1@' Makefile
```

Assurez-vous que les pages de manuel s'installent au bon endroit :

```
sed -i "s@(PREFIX)/man@(PREFIX)/share/man@g" Makefile
```

Préparez la compilation de Bzip2 avec :

```
make -f Makefile-libbz2_so
make clean
```

Signification des paramètres de make :

`-f Makefile-libbz2_so`
Bzip2 sera construit en utilisant un fichier `makefile` différent, dans ce cas le fichier `Makefile-libbz2_so` qui crée une bibliothèque `libbz2.so` dynamique et lie les outils Bzip2 avec.

Compilez et testez le paquet :

```
make
```

Installez les programmes :

```
make PREFIX=/usr install
```

Installez les bibliothèques partagées :

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

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

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

Supprimez une bibliothèque statique inutile :

```
rm -fv /usr/lib/libbz2.a
```

8.7.2. Contenu de Bzip2

Programmes installés: bunzip2 (lien vers bzip2), bzcat (lien vers bzip2), bzcmp (lien vers bzdiff), bzdiff, bzegrep (lien vers bzgrep), bzfgrep (lien vers bzgrep), bzgrep, bzip2, bzip2recover, bzless (lien vers bzmore) et bzmore
Bibliothèques installées: libbz2.so
Répertoire installé: /usr/share/doc/bzip2-1.0.8

Descriptions courtes

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

8.8. Xz-5.8.1

Le paquet Xz contient des programmes de compression et de décompression de fichiers. Il offre les possibilités des formats lzma et des formats de compression récents. La compression de fichiers textes avec **xz** donne un meilleur pourcentage de compression qu'avec les commandes **gzip** ou **bzip2** traditionnelles.

Temps de construction approximatif: 0,1 SBU
Espace disque requis: 24 Mo

8.8.1. Installation de Xz

Préparez la compilation de Xz avec :

```
./configure --prefix=/usr    \
            --disable-static \
            --docdir=/usr/share/doc/xz-5.8.1
```

Compilez le paquet :

```
make
```

Pour tester les résultats, exécutez :

```
make check
```

Installez le paquet :

```
make install
```

8.8.2. Contenu de Xz

Programmes installés: lzcat (lien vers xz), lzcmp (lien vers xzdiff), lzdiff (lien vers xzdiff), lzegrep (lien vers xzgrep), lzfgrep (lien vers xzgrep), lzgrep (lien vers xzgrep), lzless (lien vers xzless), lzma (lien vers xz), lzmadec, lzmainfo, lzmore (lien vers xzmore), unlzma (lien vers xz), unxz (lien vers xz), xz, xzcat (lien vers xz), xzcmp (lien vers xzdiff), xzdec, xzdiff, xzegrep (lien vers xzgrep), xzfgrep (lien vers xzgrep), xzgrep, xzless et xzmore
Bibliothèques installées: liblzma.so
Répertoires installés: /usr/include/lzma et /usr/share/doc/xz-5.8.1

Descriptions courtes

lzcat Décompresse vers la sortie standard
lzcmp Lance **cmp** sur des fichiers LZMA compressés
lzdiff Lance **diff** sur des fichiers LZMA compressés
lzegrep Lance **egrep** sur des fichiers LZMA compressés
lzfgrep Lance **fgrep** sur des fichiers LZMA compressés
lzgrep Lance **grep** sur des fichiers LZMA compressés
lzless Lance **less** sur des fichiers LZMA compressés
lzma Compresse ou décompresse des fichiers en utilisant le format LZMA
lzmadec Un décodeur petit et rapide pour des fichiers LZMA compressés
lzmainfo Affiche les informations contenues dans l'en-tête du fichier LZMA compressé
lzmore Lance **more** sur des fichiers LZMA compressés
unlzma Décompresse des fichiers en utilisant le format LZMA

unxz Décompresse des fichiers en utilisant le format XZ

xz Compresse ou décompresse des fichiers en utilisant le format XZ

xzcat Décompresse vers la sortie standard

xzcmp Lance **cmp** sur des fichiers Xz compressés

xzdec Un décodeur petit et rapide pour des fichiers compressés XZ

xzdiff Lance **diff** sur des fichiers XZ compressés

xzegrep Lance **egrep** sur des fichiers XZ compressés

xzfgrep Lance **fgrep** sur des fichiers XZ compressés

xzgrep Lance **grep** sur des fichiers XZ compressés

xzless Lance **less** sur des fichiers XZ compressés

xzmore Lance **more** sur des fichiers XZ compressés

`liblzma` La bibliothèque qui implémente la compression sans perte, de données rangées par blocs, utilisant les algorithmes de la chaîne Lempel-Ziv-Markov

8.9. Lz4-1.10.0

Lz4 est un algorithme de compression sans perte qui possède une vitesse de compression supérieure à 500 Mo/s par cœur. Il possède un décodeur extrêmement rapide, à la vitesse de plusieurs Go/s par cœur. Lz4 peut fonctionner avec Zstandard pour permettre aux deux algorithmes de compresser les données plus vite.

Temps de construction approximatif: 0,1 SBU
Espace disque requis: 4,2 Mo

8.9.1. Installation de Lz4

Compilez le paquet :

```
make BUILD_STATIC=no PREFIX=/usr
```

Pour tester les résultats, exécutez :

```
make -j1 check
```

Installez le paquet :

```
make BUILD_STATIC=no PREFIX=/usr install
```

8.9.2. Contenu de Lz4

Programmes installés: lz4, lz4c (lien vers lz4), lz4cat (lien vers lz4) et unlz4 (lien vers lz4)
Bibliothèque installée: liblz4.so

Descriptions courtes

lz4 Compresse ou décompresse des fichiers au format LZ4

lz4c Compresse des fichiers au format LZ4

lz4cat Liste le contenu d'un fichier compressé au format LZ4

unlz4 Décompresse des fichiers au format LZ4

`liblz4` La bibliothèque qui implémente la compression de données sans perte avec l'algorithme LZ4

8.10. Zstd-1.5.7

Zstandard est un algorithme de compression en temps réel qui fournit des ratios de compression élevés. Il propose une très large gamme de rapports entre compression et vitesse tout en étant soutenu par un décodeur très rapide.

Temps de construction approximatif: 0,4 SBU
Espace disque requis: 86 Mo

8.10.1. Installation de Zstd

Compilez le paquet :

```
make prefix=/usr
```

Note

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

Pour tester les résultats, exécutez :

```
make check
```

Installez le paquet :

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

Supprimez la bibliothèque statique :

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

8.10.2. Contenu de Zstd

Programmes installés: zstd, zstdcat (lien vers zstd), zstdgrep, zstdless, zstdmt (lien vers zstd) et unzstd (lien vers zstd)
Bibliothèque installée: libzstd.so

Descriptions courtes

zstd Compresse ou décompresse des fichiers avec le format ZSTD
zstdgrep Lance **grep** sur des fichiers compressés avec ZSTD
zstdless Lance **less** sur des fichiers compressés avec ZSTD
`libzstd` La bibliothèque implémentant la compression de données sans perte, avec l'algorithme ZSTD

8.11. File-5.46

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

Temps de construction approximatif: moins de 0,1 SBU
Espace disque requis: 19 Mo

8.11.1. Installation de File

Préparez la compilation de File :

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

Compilez le paquet :

```
make
```

Pour tester les résultats, exécutez :

```
make check
```

Installez le paquet :

```
make install
```

8.11.2. Contenu de File

Programmes installés: file
Bibliothèque installée: libmagic.so

Descriptions courtes

file Tente de classifier chaque fichier donné. Il réalise ceci grâce à l'exécution de différents tests : tests sur le système de fichiers, tests des nombres magiques et tests de langages

`libmagic` Contient des routines pour la reconnaissance de nombres magiques que le programme **file** utilise

8.12. Readline-8.3

Le paquet Readline est un ensemble de bibliothèques qui offrent des fonctionnalités d'édition des lignes de commande et des historiques.

Temps de construction approximatif: moins de 0,1 SBU
Espace disque requis: 17 Mo

8.12.1. Installation de Readline

La réinstallation de Readline aura pour conséquence que les vieilles bibliothèques seront déplacées vers <nom_bibliotheque>.old. Même si cela n'est pas normalement un problème, cela peut dans certains cas provoquer un bogue de lien dans **ldconfig**. Cela peut être évité en effectuant les deux seds suivants :

```
sed -i '/MV.*old/d' Makefile.in
sed -i '/{OLDSUFF}/c:' support/shlib-install
```

Évitez que des chemins de recherches des bibliothèques (rpath) ne soient codés en dur dans les bibliothèques partagées. Ce paquet n'a pas besoin de rpath pour une installation dans un emplacement standard et les rpath peuvent parfois causer des effets non-désirés voire des problèmes de sécurité :

```
sed -i 's/-Wl,-rpath,[^ ]*//' support/shobj-conf
```

Préparez la compilation de Readline :

```
./configure --prefix=/usr    \
            --disable-static \
            --with-curses    \
            --docdir=/usr/share/doc/readline-8.3
```

Voici la signification de la nouvelle option de configuration :

`--with-curses`
Cette option dit à Readline qu'il peut trouver les fonctions de la bibliothèque termcap dans la bibliothèque curses, au lieu d'une bibliothèque termcap séparée. Elle permet aussi de générer un fichier `readline.pc` correct.

Compilez le paquet :

```
make SHLIB_LIBS="-lncursesw"
```

Voici la signification de l'option de make :

`SHLIB_LIBS="-lncursesw"`
Cette option force Readline à se lier à la bibliothèque `libncursesw`. Pour les détails, consulter la section « Bibliothèques partagées » du fichier `README` du paquet.

Ce paquet n'a pas de suite de tests.

Installez le paquet :

```
make install
```

Si vous le souhaitez, installez la documentation :

```
install -v -m644 doc/*.{ps,pdf,html,dvi} /usr/share/doc/readline-8.3
```

8.12.2. Contenu de Readline

Bibliothèques installées: libhistory.so et libreadline.so
Répertoires installés: /usr/include/readline et /usr/share/doc/readline-8.3

Descriptions courtes

`libhistory`	Fournit une interface utilisateur cohérente pour rappeler des lignes dans l'historique
`libreadline`	Fournit un ensemble de commandes pour manipuler du texte entré dans une session interactive d'un programme

8.13. M4-1.4.20

Le paquet M4 contient un processeur de macros.

Temps de construction approximatif: 0,4 SBU
Espace disque requis: 60 Mo

8.13.1. Installation de M4

Préparez la compilation de M4 :

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

Compilez le paquet :

```
make
```

Pour tester les résultats, exécutez :

```
make check
```

Installez le paquet :

```
make install
```

8.13.2. Contenu de M4

Programme installé: m4

Descriptions courtes

m4 Copie les fichiers donnés tout en développant les macros qu'ils contiennent. Ces macros sont soit internes soit définies par l'utilisateur et peuvent prendre un nombre illimité d'arguments. En plus de la simple expansion de macros, **m4** dispose de fonctions pour inclure des fichiers nommés, lancer des commandes Unix, faire des opérations arithmétiques, manipuler du texte, pour la récursion et ainsi de suite. Le programme **m4** peut être utilisé soit comme interface d'un compilateur soit comme évaluateur de macros à part

8.14. Bc-7.0.3

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

Temps de construction approximatif: moins de 0,1 SBU
Espace disque requis: 7,8 Mo

8.14.1. Installation de Bc

Préparez la compilation de Bc :

```
CC='gcc -std=c99' ./configure --prefix=/usr -G -O3 -r
```

Voici la signification des options de configuration :

`CC='gcc -std=c99'`
Ce paramètre spécifie le compilateur et le standard C à utiliser.

`-G`
Élimine certaines parties de la suite de tests qui ne fonctionnent pas sans une version installée de GNU bc.

`-O3`
Spécifie le niveau d'optimisation à utiliser.

`-r`
Active l'utilisation de Readline pour améliorer la fonction d'édition de ligne de bc.

Compilez le paquet :

```
make
```

Pour tester bc, lancez :

```
make test
```

Installez le paquet :

```
make install
```

8.14.2. Contenu de Bc

Programmes installés: bc et dc

Descriptions courtes

bc Une calculatrice en ligne de commandes

dc Une calculatrice en ligne de commande en notation polonaise inverse

8.15. Flex-2.6.4

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

Temps de construction approximatif: 0,1 SBU
Espace disque requis: 33 Mo

8.15.1. Installation de Flex

Préparez la compilation de Flex :

```
./configure --prefix=/usr \
            --docdir=/usr/share/doc/flex-2.6.4 \
            --disable-static
```

Compilez le paquet :

```
make
```

Pour tester les résultats, exécutez :

```
make check
```

Installez le paquet :

```
make install
```

Quelques programmes ne connaissent pas encore **flex** et essaient de lancer son prédécesseur, **lex**. Pour aider ces programmes, créez un lien symbolique nommé `lex` qui lance `flex` en mode d'émulation **lex** et créez également la page de manuel ed **lex** avec un lien symbolique :

```
ln -sv flex   /usr/bin/lex
ln -sv flex.1 /usr/share/man/man1/lex.1
```

8.15.2. Contenu de Flex

Programmes installés: flex, flex++ (lien vers flex), et lex (lien vers flex)
Bibliothèques installées: libfl.so
Répertoire installé: /usr/share/doc/flex-2.6.4

Descriptions courtes

flex Un outil pour générer des programmes qui reconnaissent des motifs dans un texte. Cela permet une grande adaptabilité lors du choix des règles de recherche de motifs, ce qui élimine ainsi le besoin de développer un programme spécialisé

flex++ Une extension de flex utilisée pour générer du code et des classes C++. C'est un lien symbolique vers **flex**

lex Un lien symbolique qui exécute **flex** en mode d'émulation **lex**

`libfl` La bibliothèque `flex`

8.16. Tcl-8.6.16

Le paquet Tcl contient Tool Command Language, un langage de script robuste et polyvalent. Le paquet Expect est écrit en Tcl (prononcé [tickle]).

Temps de construction approximatif: 3,0 SBU
Espace disque requis: 91 Mo

8.16.1. Installation de Tcl

Ce paquet et les deux suivants (Expect et DejaGNU) sont installés pour prendre en charge le lancement des suites de tests de Binutils, GCC et d'autres paquets. Installer trois paquets pour effectuer des tests peut sembler excessif, mais c'est toujours rassurant, sinon essentiel, de savoir que les outils les plus importants fonctionnent correctement.

Préparez la compilation de Tcl :

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

Voici la signification des nouveaux paramètres de configuration :

`--disable-rpath`
Ce paramètre évite de coder en dur des chemins de recherche des bibliothèques (rpath) dans les fichiers binaires exécutables et les bibliothèques partagée. Ce paquet n'a pas besoin des rpath pour une installation dans l'emplacement standard et les rpath peuvent parfois causer des effets non-désirés voire des problèmes de sécurité.

Construisez le paquet :

```
make

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

sed -e "s|$SRCDIR/unix/pkgs/tdbc1.1.10|/usr/lib/tdbc1.1.10|" \
    -e "s|$SRCDIR/pkgs/tdbc1.1.10/generic|/usr/include|"     \
    -e "s|$SRCDIR/pkgs/tdbc1.1.10/library|/usr/lib/tcl8.6|"  \
    -e "s|$SRCDIR/pkgs/tdbc1.1.10|/usr/include|"             \
    -i pkgs/tdbc1.1.10/tdbcConfig.sh

sed -e "s|$SRCDIR/unix/pkgs/itcl4.3.2|/usr/lib/itcl4.3.2|" \
    -e "s|$SRCDIR/pkgs/itcl4.3.2/generic|/usr/include|"    \
    -e "s|$SRCDIR/pkgs/itcl4.3.2|/usr/include|"            \
    -i pkgs/itcl4.3.2/itclConfig.sh

unset SRCDIR
```

Les diverses instructions « sed » après la commande « make » suppriment des références au répertoire de construction des fichiers de configuration et les remplacent par le répertoire d'installation. Cela n'est pas requis pour le reste de LFS, mais peut être requis pour un paquet construit plus tard avec Tcl.

Pour tester les résultats, exécutez :

```
make test
```

Installez le paquet :

```
make install
chmod 644 /usr/lib/libtclstub8.6.a
```

Rendez la bibliothèque installée réinscriptible pour que les symboles de débogages puissent être supprimés plus tard :

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

Installez les en-têtes de Tcl. Le paquet suivant, Expect, en a besoin.

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

Maintenant créez un lien symbolique nécessaire :

```
ln -sfv tclsh8.6 /usr/bin/tclsh
```

Renommez une page de manuel qui entre en conflit avec une page de manuel de Perl :

```
mv /usr/share/man/man3/{Thread,Tcl_Thread}.3
```

Éventuellement, installez la documentation en exécutant les commandes suivantes :

```
cd ..
tar -xf ../tcl8.6.16-html.tar.gz --strip-components=1
mkdir -v -p /usr/share/doc/tcl-8.6.16
cp -v -r  ./html/* /usr/share/doc/tcl-8.6.16
```

8.16.2. Contenu de Tcl

Programmes installés: tclsh (lien vers tclsh8.6) et tclsh8.6
Bibliothèque installée: libtcl8.6.so et libtclstub8.6.a

Descriptions courtes

tclsh8.6 Le shell de commande de Tcl

tclsh Un lien vers tclsh8.6

`libtcl8.6.so` La bibliothèque Tcl

`libtclstub8.6.a` La bibliothèque de base de Tcl

8.17. Expect-5.45.4

Le paquet Expect contient des outils pour automatiser, via des dialogues scriptés, des applications interactives comme **telnet**, **ftp**, **passwd**, **fsck**, **rlogin** et **tip**. Expect est aussi utile pour tester ces mêmes applications et faciliter toutes sortes de tâches qui sont trop compliquées avec quoi que ce soit d'autre. Le cadre de tests DejaGnu est écrit en Expect.

Temps de construction approximatif: 0,2 SBU
Espace disque requis: 3,9 Mo

8.17.1. Installation d'Expect

Expect a besoin de PTY pour fonctionner. Vérifiez que les PTY fonctionnent correctement dans l'environnement chroot en effectuant un simple test :

```
python3 -c 'from pty import spawn; spawn(["echo", "ok"])'
```

Cette commande devrait renvoyer `ok`. Sinon, si la sortie contient `OSError: out of pty devices`, alors l'environnement n'est pas configuré pour utiliser des PTY. Vous devrez sortir de l'environnement chroot, relire Section 7.3, « Préparer les systèmes de fichiers virtuels du noyau » et vous assurer que le système de fichiers `devpts` (et les autres systèmes de fichiers virtuels du noyau) est monté correctement. Ensuite, entrez de nouveau dans l'environnement chroot en suivant Section 7.4, « Entrer dans l'environnement chroot ». Ce problème doit être résolu avant de continuer, ou les suites de tests qui nécessitent Expect (par exemple celles de Bash, Binutils, GCC, GDBM et bien sûr Expect lui-même) échoueront de manière catastrophique et d'autres problèmes subtiles pourraient apparaître.

Maintenant, effectuez quelques changements pour permettre au paquet de construire avec gcc-15.1 ou supérieur :

```
patch -Np1 -i ../expect-5.45.4-gcc15-1.patch
```

Préparez la compilation d'Expect :

```
./configure --prefix=/usr           \
            --with-tcl=/usr/lib     \
            --enable-shared         \
            --disable-rpath         \
            --mandir=/usr/share/man \
            --with-tclinclude=/usr/include
```

Voici la signification des options de configuration :

`--with-tcl=/usr/lib`
Ce paramètre est requis pour dire à **configure** où le script **tclConfig.sh** se trouve.

`--with-tclinclude=/usr/include`
Cela dit explicitement à Expect où trouver les en-têtes internes de Tcl.

Construisez le paquet :

```
make
```

Pour tester les résultats, exécutez :

```
make test
```

Installez le paquet :

```
make install
ln -svf expect5.45.4/libexpect5.45.4.so /usr/lib
```

8.17.2. Contenu d'Expect

Programme installé: expect
Bibliothèque installée: libexpect5.45.4.so

Descriptions courtes

expect	Communique avec les autres programmes interactifs selon un script
`libexpect-5.45.4.so`	Contient des fonctions qui permettent à Expect d'être utilisé comme une extension Tcl ou directement à partir du langage C ou du langage C++ (sans Tcl)

8.18. DejaGNU-1.6.3

Le paquet DejaGnu contient un ensemble de travail pour lancer les suites de tests d'outils GNU. Il est écrit en **expect**, qui lui-même utilise Tcl (langage de commande des outils).

Temps de construction approximatif: moins de 0,1 SBU
Espace disque requis: 6,9 Mo

8.18.1. Installation de DejaGNU

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

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

Préparez la compilation de DejaGNU :

```
../configure --prefix=/usr
makeinfo --html --no-split -o doc/dejagnu.html ../doc/dejagnu.texi
makeinfo --plaintext       -o doc/dejagnu.txt  ../doc/dejagnu.texi
```

Pour tester les résultats, exécutez :

```
make check
```

Installez le paquet :

```
make install
install -v -dm755  /usr/share/doc/dejagnu-1.6.3
install -v -m644   doc/dejagnu.{html,txt} /usr/share/doc/dejagnu-1.6.3
```

8.18.2. Contenu de DejaGNU

Programme installé: dejagnu et runtest

Descriptions courtes

dejagnu Lanceur de commande auxiliaire de DejaGNU

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

8.19. Pkgconf-2.5.1

Le paquet pkgconf est le successeur de pkg-config et contient un outil pour passer le chemin d'inclusion ou de bibliothèque des outils de construction pendant les phases de configuration et de construction de l'installation des paquets.

Temps de construction approximatif: moins de 0,1 SBU
Espace disque requis: 5,0 Mo

8.19.1. Installation de Pkgconf

Préparez la compilation de Pkgconf :

```
./configure --prefix=/usr    \
            --disable-static \
            --docdir=/usr/share/doc/pkgconf-2.5.1
```

Compilez le paquet :

```
make
```

Installez le paquet :

```
make install
```

Pour maintenir la compatibilité avec le pkg-config original, créez deux liens symboliques :

```
ln -sv pkgconf   /usr/bin/pkg-config
ln -sv pkgconf.1 /usr/share/man/man1/pkg-config.1
```

8.19.2. Contenu de Pkgconf

Programmes installés: pkgconf, pkg-config (lien vers pkgconf) et bomtool
Bibliothèque installée: libpkgconf.so
Répertoire installé: /usr/share/doc/pkgconf-2.5.1

Descriptions courtes

pkgconf Renvoie les métadonnées d'une bibliothèque ou d'un paquet donné

bomtool Génère une nomenclature logiciels à partir des fichiers .pk de pkg-config

`libpkgconf` Contient la plupart des fonctionnalités de pkgconf, tout en permettant à d'autres outils comme des IDE et des compilateurs de l'utiliser dans leur cadre

8.20. Binutils-2.45

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

Temps de construction approximatif: 1,6 SBU
Espace disque requis: 832 Mo

8.20.1. Installation de Binutils

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

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

Préparez la compilation de Binutils :

```
../configure --prefix=/usr       \
             --sysconfdir=/etc   \
             --enable-ld=default \
             --enable-plugins    \
             --enable-shared     \
             --disable-werror    \
             --enable-64-bit-bfd \
             --enable-new-dtags  \
             --with-system-zlib  \
             --enable-default-hash-style=gnu
```

Voici la signification des nouveaux paramètres de configuration :

`--enable-ld=default`
Construit l'éditeur de liens bfd original et l'installe à la fois en tant que ld (l'éditeur par défaut) et ld.bfd.

`--enable-plugins`
Permet la prise en charge des plugins pour l'éditeur de lien.

`--with-system-zlib`
Utilise la version déjà installée de la bibliothèque zlib au lieu de construire la version inclue.

Compilez le paquet :

```
make tooldir=/usr
```

Signification des paramètres de make :

`tooldir=/usr`
Normalement, le nom du répertoire tooldir (où seront situés les exécutables) est configuré de cette manière : `$(exec_prefix)/$(target_alias)`. Les machines x86_64 y ajouteront `/usr/x86_64-pc-linux-gnu`. Comme il s'agit d'un système personnalisé, il n'est pas nécessaire d'avoir un répertoire spécifique à la cible dans `/usr`. Si le système était utilisé pour la compilation croisée (par exemple pour compiler un paquet sur une machine Intel qui génère du code pouvant être exécuté sur des machines PowerPC), le répertoire s'appellerait `$(exec_prefix)/$(target_alias)`.

Important

La suite de tests de Binutils est indispensable.Ne l'oubliez sous aucun prétexte.

Testez les résultats :

```
make -k check
```

Pour afficher la liste des tests qui ont échoué, exécutez :

```
grep '^FAIL:' $(find -name '*.log')
```

Installez le paquet :

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

Supprimez les bibliothèques statiques inutiles et d'autres fichiers inutiles :

```
rm -rfv /usr/lib/lib{bfd,ctf,ctf-nobfd,gprofng,opcodes,sframe}.a \
        /usr/share/doc/gprofng/
```

8.20.2. Contenu de Binutils

Programmes installés: addr2line, ar, as, c++filt, dwp, elfedit, gprof, gprofng, ld, ld.bfd, nm, objcopy, objdump, ranlib, readelf, size, strings et strip
Bibliothèques installées: libbfd.so, libctf.so, libctf-nobfd.so, libgprofng.so, libopcodes.so et libsframe.so
Répertoire installé: /usr/lib/ldscripts

Descriptions courtes

addr2line Traduit les adresses de programmes en noms de fichier et numéros de ligne ; en fonction de l'adresse et du nom de l'exécutable, il utilise les informations de débogage disponibles dans l'exécutable pour déterminer le fichier source et le numéro de ligne associés à cette adresse

ar Crée, modifie et extrait des archives

as Assemble la sortie de la commande **gcc** en fichiers objet

c++filt Utilisé par l'éditeur de liens pour récupérer les symboles C++ et Java, et pour empêcher les fonctions surchargées d'entrer en conflit

dwp L'utilitaire d'empaquetage DWARF

elfedit Met à jour l'en-tête ELF des fichiers ELF

gprof Affiche les données de profil du graphe d'appel

gprofng Récupère et analyse les données de performance

ld Un éditeur de liens qui combine un certain nombre d'objets et de fichiers d'archive en un seul fichier, en déplaçant leurs données et en regroupant les références des symboles

ld.bfd Lien physique vers **ld**

nm Liste les symboles présents dans un fichier objet donné

objcopy Traduit un type de fichier objet en un autre

objdump Affiche les informations concernant un fichier objet donné, avec des options permettant de contrôler les données à afficher. Ces données sont surtout utiles aux programmeurs qui travaillent sur les outils de compilation

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

readelf Affiche des informations sur les binaires de type ELF

size Liste la taille de la section et la taille totale des fichiers objet donnés

strings Affiche pour chaque fichier donné la séquence de caractères imprimables qui sont d'au moins la taille spécifiée (quatre par défaut). Pour les fichiers objet, cette commande affiche par défaut uniquement les chaînes des sections d'initialisation et de chargement, et parcourt le fichier entier pour les autres types de fichiers

strip	Supprime les symboles des fichiers objet
`libbfd`	Bibliothèque Binary File Descriptor
`libctf`	Bibliothèque de prise en charge du débogage du format compatible ANSI C
`libctf-nobfd`	Une variante de libctf qui n'utilise pas la fonctionnalité libbfd
`libgprofng`	Une bibliothèque contenant la plupart des routines utilisées par **gprofng**
`libopcodes`	Une bibliothèque de gestion des codes opération, la « version lisible » des instructions du processeur. Elle est utilisée pour construire des utilitaires comme **objdump**
`libsframe`	Une bibliothèque pour prendre en charge le bactracking en ligne avec un simple dérouleur

8.21. GMP-6.3.0

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

Temps de construction approximatif: 0,3 SBU
Espace disque requis: 54 Mo

8.21.1. Installation de GMP

Note

Si vous construisez pour un x86 32 bits, mais si vous avez un processeur capable d'exécuter du code 64 bits *et* si vous avez spécifié `CFLAGS` dans l'environnement, le script configure va essayer de configurer pour du 64 bits et va échouer. Évitez cela en invoquant la commande configure ci-dessous avec

```
ABI=32 ./configure ...
```

Note

Les paramètres par défaut de GMP produisent des bibliothèques optimisées pour le processeur de l'hôte. Si vous souhaitez obtenir des bibliothèques convenables pour des processeurs moins puissants, vous pouvez créer des bibliothèques génériques en ajoutant l'option `>--host=none-linux-gnu` à la commande **configure**.

Tout d'abord, ajustez le paquet pour le rendre compatible avec gcc-15 et supérieur :

```
sed -i '/long long t1;/,+1s/()/(...)/' configure
```

Préparez la compilation de GMP :

```
./configure --prefix=/usr    \
            --enable-cxx     \
            --disable-static \
            --docdir=/usr/share/doc/gmp-6.3.0
```

Voici la signification des nouvelles options de configure :

`--enable-cxx`
Ce paramètre active la prise en charge de C++

`--docdir=/usr/share/doc/gmp-6.3.0`
Cette variable indique le bon emplacement de la documentation.

Compilez le paquet et générez la documentation HTML :

```
make
make html
```

Important

La suite de tests de GMP dans cette section est considérée comme critique. Ne la sautez en aucun cas.

Testez les résultats :

```
make check 2>&1 | tee gmp-check-log
```

Attention

Le code de gmp est hautement optimisé pour le processeur sur lequel il est construit. Parfois, le code chargé de détecter le processeur identifie mal les capacités du système et produira des erreurs dans les tests ou d'autres applications utilisant les bibliothèques gmp avec le message `Illegal instruction`. Dans ce cas, gmp devrait être reconfiguré avec l'option `--host=none-linux-gnu` et reconstruit.

Assurez-vous qu'au moins 199 tests de la suite de tests réussissent tous. Vérifiez les résultats en exécutant la commande suivante :

```
awk '/# PASS:/{total+=$3} ; END{print total}' gmp-check-log
```

Installez le paquet et sa documentation :

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

8.21.2. Contenu de GMP

Bibliothèques installées: libgmp.so et libgmpxx.so
Répertoire installé: /usr/share/doc/gmp-6.3.0

Descriptions courtes

`libgmp` Contient les fonctions de maths de précision

`libgmpxx` Contient des fonctions de maths de précision pour C++

8.22. MPFR-4.2.2

Le paquet MPFR contient des fonctions d'arithmétique multi-précision.

Temps de construction approximatif: 0,2 SBU
Espace disque requis: 43 Mo

8.22.1. Installation de MPFR

Préparez la compilation de MPFR :

```
./configure --prefix=/usr        \
            --disable-static     \
            --enable-thread-safe \
            --docdir=/usr/share/doc/mpfr-4.2.2
```

Compilez le paquet et générez la documentation HTML :

```
make
make html
```

Important

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

Testez les résultats et assurez-vous que les 198 tests sont réussis :

```
make check
```

Installez le paquet et sa documentation :

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

8.22.2. Contenu de MPFR

Bibliothèques installées: libmpfr.so
Répertoire installé: /usr/share/doc/mpfr-4.2.2

Descriptions courtes

`libmpfr` Contient des fonctions arithmétique multi-précision

8.23. MPC-1.3.1

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

Temps de construction approximatif: 0,1 SBU
Espace disque requis: 22 Mo

8.23.1. Installation de MPC

Préparez la compilation de MPC :

```
./configure --prefix=/usr    \
            --disable-static \
            --docdir=/usr/share/doc/mpc-1.3.1
```

Compilez le paquet et générez la documentation HTML :

```
make
make html
```

Pour tester les résultats, exécutez :

```
make check
```

Installez le paquet et sa documentation :

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

8.23.2. Contenu de MPC

Bibliothèques installées: libmpc.so
Répertoire installé: /usr/share/doc/mpc-1.3.1

Descriptions courtes

`libmpc` Contient des fonctions mathématiques complexes

8.24. Attr-2.5.2

Le paquet attr contient des outils d'administration des attributs étendus des objets du système de fichier.

Temps de construction approximatif: moins de 0,1 SBU
Espace disque requis: 4,1 Mo

8.24.1. Installation d'Attr

Préparez la compilation d'Attr :

```
./configure --prefix=/usr     \
            --disable-static  \
            --sysconfdir=/etc \
            --docdir=/usr/share/doc/attr-2.5.2
```

Compilez le paquet :

```
make
```

Il faut lancer les tests sur un système de fichiers qui prend en charge les attributs étendus, comme les systèmes de fichiers ext2, ext3, ou ext4. Pour tester les résultats, lancez :

```
make check
```

Installez le paquet :

```
make install
```

8.24.2. Contenu d'Attr

Programmes installés: attr, getfattr et setfattr
Bibliothèque installée: libattr.so
Répertoires installés: /usr/include/attr et /usr/share/doc/attr-2.5.2

Descriptions courtes

attr Étend les attributs des objets d'un système de fichiers

getfattr Affiche les attributs étendus des objets d'un système de fichiers

setfattr Définit les attributs étendus des objets d'un système de fichiers

`libattr` Contient la bibliothèque de fonctions pour la manipulation des attributs étendus

8.25. Acl-2.3.2

Le paquet Acl contient des outils d'administration des Access Control Lists (listes de contrôle d'accès) qui sont utilisés pour définir des droits d'accès discrétionnaires fins aux fichiers et aux répertoires.

Temps de construction approximatif: moins de 0,1 SBU
Espace disque requis: 6,5 Mo

8.25.1. Installation d'Acl

Préparez la compilation d'Acl :

```
./configure --prefix=/usr    \
            --disable-static \
            --docdir=/usr/share/doc/acl-2.3.2
```

Compilez le paquet :

```
make
```

Les tests d'Acl doivent être exécutés sur un système de fichiers qui prend en charge les contrôles d'accès. Pour tester les résultats, exécutez :

```
make check
```

Un test nommé `test/cp.test` est connu pour échouer parce que Coreutils n'est pas encore construit avec la prise en charge d'Acl.

Installez le paquet :

```
make install
```

8.25.2. Contenu d'Acl

Programmes installés: chacl, getfacl et setfacl
Bibliothèque installée: libacl.so
Répertoires installés: /usr/include/acl et /usr/share/doc/acl-2.3.2

Descriptions courtes

chacl Modifie la liste de contrôle d'accès d'un fichier ou d'un répertoire

getfacl Donne les listes de contrôle des accès à un fichier

setfacl Définit les listes de contrôle d'accès à un fichier

`libacl` Contient la bibliothèque de fonction pour la manipulation de Access Control Lists

8.26. Libcap-2.76

Le paquet Libcap implémente les interfaces du niveau utilisateur avec les fonctions POSIX 1003.1e disponibles dans les noyaux Linux. Ces possibilités établissent le partage des pouvoirs des privilèges root à un ensemble de droits distincts.

Temps de construction approximatif: moins de 0,1 SBU
Espace disque requis: 3,1 Mo

8.26.1. Installation de Libcap

Évitez que des bibliothèques statiques ne soient installées :

```
sed -i '/install -m.*STA/d' libcap/Makefile
```

Compilez le paquet :

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

Voici la signification de l'option de make :

`lib=lib`
Ce paramètre fait en sorte que la bibliothèque soit installée dans `/usr/lib` plutôt que dans `/usr/lib64` sur x86_64. Il n'a aucun effet sur x86.

Pour tester les résultats, exécutez :

```
make test
```

Installez le paquet :

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

8.26.2. Contenu de Libcap

Programmes installés: capsh, getcap, getpcaps, et setcap
Bibliothèque installée: libcap.so et libpsx.so

Descriptions courtes

capsh Une enveloppe shell pour voir et contraindre la prise en charge de ces capacités
getcap Examine les capacités d'un fichier
getpcaps Affiche les capacités des processus requis
setcap Définit les capacités d'un fichier
`libcap` Contient les fonctions de la bibliothèque de manipulation des capacités POSIX 1003.1e
`libpsx` Contient des fonctions pour la prise en charge de la sémantique POSIX des appels systèmes associés avec la bibliothèque pthread

8.27. Libxcrypt-4.4.38

Le paquet Libxcrypt contient une bibliothèque moderne pour le hashage en sens unique des mots de passe.

Temps de construction approximatif: 0,1 SBU
Espace disque requis: 12 Mo

8.27.1. Installation de Libxcrypt

Préparez la compilation de Libxcrypt :

```
./configure --prefix=/usr                \
            --enable-hashes=strong,glibc \
            --enable-obsolete-api=no     \
            --disable-static             \
            --disable-failure-tokens
```

Voici la signification des nouvelles options de configure :

`--enable-hashes=strong,glibc`
Construit les algorithmes de hashage forts recommandés pour les cas d'usages sécuritaires, et les algorithmes de hashage fournis par la `libcrypt` traditionnelle de Glibc pour la compatibilité.

`--enable-obsolete-api=no`
Désactive les fonctions obsolètes de l'API. Elles ne sont pas requises pour un système Linux moderne construit à partir des sources.

`--disable-failure-tokens`
Désactive la fonctionnalité de jeton d'échec. Cela est requis pour la compatibilité avec les bibliothèques de hashage traditionnelles de certaines plateformes, mais un système Linux basé sur Glibc n'en a pas besoin.

Compilez le paquet :

```
make
```

Pour tester les résultats, exécutez :

```
make check
```

Installez le paquet :

```
make install
```

Note

Les instructions ci-dessus désactivent les fonctions obsolètes de l'API car aucun paquet installé en le compilant à partir des sources ne se lieraient à elles à l'exécution. Cependant, les seules applications disponibles uniquement au format binaire connues qui se lient à ces fonctions nécessitent l'ABI version 1. Si vous devez avoir ces fonctions à cause d'une application binaire ou pour être compatible avec la LSB, construisez de nouveau le paquet avec les commandes suivantes :

```
make distclean
./configure --prefix=/usr                \
            --enable-hashes=strong,glibc \
            --enable-obsolete-api=glibc  \
            --disable-static             \
            --disable-failure-tokens
make
cp -av --remove-destination .libs/libcrypt.so.1* /usr/lib
```

8.27.2. Contenu de Libxcrypt

Bibliothèques installées: libcrypt.so

Descriptions courtes

`libcrypt` Contient des fonctions pour hasher des mots de passe

8.28. Shadow-4.18.0

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

Temps de construction approximatif: 0,1 SBU
Espace disque requis: 115 Mo

8.28.1. Installation de Shadow

Important

Si vous avez installé Linux-PAM, vous devriez suivre *les instructions BLFS* au lieu de cette page pour construire (ou reconstruire ou mettre à jour) shadow.

Note

Si vous souhaitez forcer l'utilisation de mots de passe forts, commencez par *installer et configurer Linux-PAM*. Ensuite, *installez et configurez shadow avec la prise en charge de PAM*. Enfin, *installez libpwquality et configurez PAM pour l'utiliser*.

Désactivez l'installation du programme **groups** et de ses pages de manuel car Coreutils en fournit une meilleure version. Cela empêche aussi l'installation de pages de manuel déjà installées dans Section 8.3, « Man-pages-6.15 » :

```
sed -i 's/groups$(EXEEXT) //' src/Makefile.in
find man -name Makefile.in -exec sed -i 's/groups\.1 / /'   {} \;
find man -name Makefile.in -exec sed -i 's/getspnam\.3 / /' {} \;
find man -name Makefile.in -exec sed -i 's/passwd\.5 / /'   {} \;
```

Au lieu d'utiliser la méthode *crypt* par défaut, utilisez la méthode *YESCRYPT* de chiffrement de mot de passe bien plus sécurisée, qui autorise aussi les mots de passe plus longs que huit caractères. Il est également nécessaire de changer l'emplacement obsolète de `/var/spool/mail` pour les boîtes e-mail de l'utilisateur que Shadow utilise par défaut en l'endroit `/var/mail` utilisé actuellement. Ensuite, retirez `/bin` et `/sbin` de `PATH`, car ce sont de simples liens symboliques vers leur contrepartie dans `/usr`.

Avertissement

Ajouter `/bin` ou `/sbin` dans la variable `PATH` peut causer l'échec de la construction de certains paquets de BLFS, donc ne le faites pas dans le fichier `.bashrc` ni nul part ailleurs.

```
sed -e 's:#ENCRYPT_METHOD DES:ENCRYPT_METHOD YESCRYPT:' \
    -e 's:/var/spool/mail:/var/mail:'                   \
    -e '/PATH=/{s@/sbin:@@;s@/bin:@@}'                  \
    -i etc/login.defs
```

Préparez la compilation de Shadow :

```
touch /usr/bin/passwd
./configure --sysconfdir=/etc   \
            --disable-static    \
            --with-{b,yes}crypt \
            --without-libbsd    \
            --with-group-name-max-length=32
```

Voici la signification des nouvelles options de configuration :

touch /usr/bin/passwd
Le fichier `/usr/bin/passwd` a besoin d'exister parce que son emplacement est codé en dur dans certains programmes. S'il n'existe pas, le script d'installation va en créer un par défaut au mauvais endroit.

--with-{b,yes}crypt
: Le shell étend ce paramètre en deux paramètres, *--with-bcrypt* et *--with-yescrypt*. Ils permettent à shadow d'utiliser les algorithmes Bcrypt et Yescrypt implémentés par Libxcrypt pour hasher les mots de passe. Ces algorithmes sont plus sécurisés (en particulier, bien plus résistants aux attaques basés sur un GPU) que les algorithmes SHA traditionnels.

--with-group-name-max-length=32
: La longueur maximum d'un nom d'utilisateur est de 32 caractères. Le paramètre règle un plafond similaire pour les noms de groupes.

--without-libbsd
: Évite d'utiliser la foction readpassphrase de libbsd qui ne fait pas partie de LFS. Cela permet d'utiliser une copie interne à la place.

Compilez le paquet :

```
make
```

Ce paquet n'a pas de suite de tests.

Installez le paquet :

```
make exec_prefix=/usr install
make -C man install-man
```

8.28.2. Configuration de Shadow

Ce paquet contient des outils pour ajouter, modifier, supprimer des utilisateurs et des groupes, initialiser et changer leur mot de passe, et bien d'autres tâches administratives. Pour une explication complète de ce que signifie *password shadowing*, jetez un œil dans le fichier `doc/HOWTO` à l'intérieur du répertoire source. Il reste une chose à garder à l'esprit si vous décidez d'utiliser le support de Shadow : les programmes qui ont besoin de vérifier les mots de passe (gestionnaires d'affichage, programmes FTP, démons pop3 , etc.) ont besoin d'être compatibles avec shadow, c'est-à-dire qu'ils ont besoin d'être capables de fonctionner avec des mots de passe shadow.

Pour activer les mots de passe shadow, lancez la commande suivante :

```
pwconv
```

Pour activer les mots de passe shadow pour les groupes, lancez :

```
grpconv
```

La configuration par défaut de Shadow pour l'outil **useradd** présente quelques inconvénients qui appellent quelques explications. Tout d'abord, l'action par défaut de l'outil **useradd** est de créer un utilisateur et un groupe du même nom que l'utilisateur. Par défaut les numéros d'ID utilisateur (UID) et d'ID de groupe (GID) commenceront à 1000. Cela signifie que si vous ne passez pas de paramètres à **useradd**, chaque utilisateur sera membre d'un groupe unique sur le système. Si vous ne désirez pas ce comportement, vous devrez passer le paramètre *-g* ou *-N* à **useradd** ou changer le paramètre *USERGROUPS_ENAB* dans `/etc/login.defs`. Voir *useradd(8)* pour plus d'informations.

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

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

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

GROUP=999
: Ce paramètre initialise le début des numéros de groupe utilisés dans le fichier `/etc/group`. La valeur 999 particulière provient du paramètre *--gid* ci-dessus. Vous pouvez le remplacer par la valeur de votre choix.

Remarquez que **useradd** ne réutilisera jamais un UID ou un GID. Si le numéro identifié dans ce paramètre est déjà utilisé, il utilisera le numéro disponible suivant celui-ci. Remarquez aussi que si vous n'avez pas de groupe de GID égal à ce numéro sur votre système la première fois que vous utilisez **useradd** sans le paramètre *-g*, vous obtiendrez un message d'erreur sur le terminal qui dit : `useradd: unknown GID 999`, bien que le compte soit correctement créé. C'est pourquoi nous avons créé le groupe `users` avec cet identifiant de groupe dans le Section 7.6, « Création des fichiers et des liens symboliques essentiels. »

`CREATE_MAIL_SPOOL=yes`

Il résulte de ce paramètre que **useradd** crée un fichier de boîte mail pour chaque nouvel utilisateur. **useradd** rendra le groupe `mail` propriétaire de ce fichier avec les droits 0660. Si vous préférez ne pas créer ces fichiers, exécutez la commande suivante :

```
sed -i '/MAIL/s/yes/no/' /etc/default/useradd
```

8.28.3. Configurer le mot de passe de root

Choisissez un mot de passe pour l'utilisateur *root* et configurez-le avec :

```
passwd root
```

8.28.4. Contenu de Shadow

Programmes installés: chage, chfn, chgpasswd, chpasswd, chsh, expiry, faillog, getsubids, gpasswd, groupadd, groupdel, groupmems, groupmod, grpck, grpconv, grpunconv, login, logoutd, newgidmap, newgrp, newuidmap, newusers, nologin, passwd, pwck, pwconv, pwunconv, sg (lien vers newgrp), su, useradd, userdel, usermod, vigr (lien vers vipw) et vipw
Répertoires installés: /etc/default et /usr/include/shadow
Bibliothèques installées: libsubid.so

Descriptions courtes

chage Utilisé pour modifier le nombre maximum de jours entre des modifications obligatoires du mot de passe

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

chgpasswd Utilisé pour mettre à jour des mots de passe en lot

chpasswd Utilisé pour mettre à jour les mots de passe utilisateurs en lot

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

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

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

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

gpasswd Est utilisé pour ajouter et supprimer des membres et des administrateurs aux groupes

groupadd Crée un groupe avec le nom donné

groupdel Supprime le groupe ayant le nom donné

groupmems Permet à un utilisateur d'administrer la liste des membres de son groupe sans avoir besoin des privilèges du super utilisateur.

groupmod Est utilisé pour modifier le nom ou le GID du groupe

grpck Vérifie l'intégrité des fichiers `/etc/group` et `/etc/gshadow`

grpconv Crée ou met à jour le fichier shadow à partir du fichier group standard

grpunconv Met à jour `/etc/group` à partir de `/etc/gshadow` puis supprime ce dernier

login Est utilisé par le système pour permettre aux utilisateurs de se connecter

logoutd Est un démon utilisé pour renforcer les restrictions sur les temps et ports de connexion

newgidmap Est utilisé pour configurer la correspondance des gid d'un espace de nom utilisateur

newgrp Est utilisé pour modifier le GID courant pendant une session de connexion

newuidmap Est utilisé pour configurer la correspondance des uid d'un espace de nom utilisateur

newusers Est utilisé pour créer ou mettre à jour toute une série de comptes utilisateur en une fois

nologin Affiche un message selon lequel un compte n'est pas disponible. Destiné à être utilisé comme shell par défaut pour des comptes qui ont été désactivés

passwd Est utilisé pour modifier le mot de passe d'un utilisateur ou d'un groupe

pwck Vérifie l'intégrité des fichiers de mots de passe, `/etc/passwd` et `/etc/shadow`

pwconv Crée ou met à jour le fichier de mots de passe shadow à partir du fichier password habituel

pwunconv Met à jour `/etc/passwd` à partir de `/etc/shadow` puis supprime ce dernier

sg Exécute une commande donnée lors de l'initialisation du GID de l'utilisateur à un groupe donné

su Lance un shell en substituant les ID de l'utilisateur et du groupe

useradd Crée un nouvel utilisateur avec le nom donné ou met à jour les informations par défaut du nouvel utilisateur

userdel Supprime le compte utilisateur indiqué

usermod Est utilisé pour modifier le nom de connexion de l'utilisateur, son UID (*User Identification*, soit Identification Utilisateur), shell, groupe initial, répertoire personnel, etc.

vigr Édite les fichiers `/etc/group` ou `/etc/gshadow`

vipw Édite les fichiers `/etc/passwd` ou `/etc/shadow`

`libsubid` bibliothèque de traitement des intervalles subordonnés d'ID des utilisateurs et des groupes

8.29. GCC-15.2.0

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

Temps de construction approximatif: 46 SBU (avec les tests)

Espace disque requis: 6,6 Go

8.29.1. Installation de GCC

Si vous construisez sur x86_64, changez le nom du répertoire par défaut des bibliothèques 64 bits en « lib » :

```
case $(uname -m) in
  x86_64)
    sed -e '/m64=/s/lib64/lib/' \
        -i.orig gcc/config/i386/t-linux64
  ;;
esac
```

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

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

Préparez la compilation de GCC :

```
../configure --prefix=/usr            \
             LD=ld                    \
             --enable-languages=c,c++ \
             --enable-default-pie     \
             --enable-default-ssp     \
             --enable-host-pie        \
             --disable-multilib       \
             --disable-bootstrap      \
             --disable-fixincludes    \
             --with-system-zlib
```

Remarquez que pour d'autres langages de programmation, il existe des pré-requis qui ne sont pas encore disponibles. Consultez la *page GCC du livre BLFS* pour des instructions sur la manière de construire tous les langages pris en charge par GCC.

Voici la signification des nouveaux paramètres de configuration :

`LD=ld`

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

`--disable-fixincludes`

Par défaut, pendant l'installation de GCC, certains en-têtes du système seraient « corrigés » pour fonctionner avec GCC. Ce n'est pas nécessaire sur un système Linux moderne, et peut être dangereux si un paquet est réinstallé après l'installation de GCC. Ce paramètre évite que GCC ne « corrige » les en-têtes.

`--with-system-zlib`

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

Note

PIE (exécutable indépendant de la position) est une technique pour produire des programmes binaires qui peuvent être chargés n'importe où en mémoire. Sans PIE, la fonctionnalité de sécurité nommée ASLR (randomisation de l'agencement de l'espace d'adressage) peut être appliquée pour les bibliothèques partagées, mais pas pour l'exécutable lui-même. Activer PIE permet l'ASLR des exécutables en plus des bibliothèques partagées et réduit certaines attaques basées sur des adresses fixes de code sensible ou de données dans les exécutables.

SSP (protection contre l'écrasement de la pile) est une technique qui s'assure que la pile des paramètres n'est pas corrompue. La corruption de pile peut par exemple changer l'adresse de retour d'une sous-routine, ce qui permettrait de transférer le contrôle à du code dangereux (qui existerait dans le programme ou les bibliothèques partagées, ou éventuellement injecté par l'attaquant) au lieu du code d'origine.

Compilez le paquet :

```
make
```

Important

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

GCC peut avoir besoin de plus d'espace en pile pour compiler certains motifs de code particulièrement complexes. Par précaution, sur les distributions qui ont une limite de pile réduite, configurez la limite de taille de pile à l'infini. Sur la plupart des distributions hôtes (et le système LFS final) la limite stricte est infinie par défaut, mais il n'y a pas d'inconvénient à l'indiquer explicitement. Il n'est pas nécessaire de limiter la limite non stricte car GCC la configurera à une valeur appropriée, tant que cette valeur ne dépasse pas la limite stricte :

```
ulimit -s -H unlimited
```

Maintenant, supprimez plusieurs échecs aux tests connus :

```
sed -e '/cpython/d' -i ../gcc/testsuite/gcc.dg/plugin/plugin.exp
```

Testez les résultats en tant qu'utilisateur non privilégié, mais ne vous arrêtez pas aux erreurs :

```
chown -R tester .
su tester -c "PATH=$PATH make -k check"
```

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

```
../contrib/test_summary
```

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

Vous pouvez comparer les résultats avec ceux situés dans *https://www.linuxfromscratch.org/lfs/build-logs/12.4/* et *https://gcc.gnu.org/ml/gcc-testresults/*.

Les tests liés à `pr90579.c` sont connus pour échouer.

Quelques échecs inattendus sont parfois inévitables. Dans certains cas les échecs des tests dépendent du matériel spécifique du système. Sauf si les résultats des tests sont très différents de ceux sur l'adresse ci-dessus, vous pouvez poursuivre en toute sécurité.

Installez le paquet :

```
make install
```

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

```
chown -v -R root:root \
    /usr/lib/gcc/$(gcc -dumpmachine)/15.2.0/include{,-fixed}
```

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

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

Beaucoup de paquets utilisent le nom **cc** pour appeler le compilateur C. Nous avons déjà créé **cc** comme un lien symbolique dans gcc-pass2, créez également sa page de manuel avec un lien symbolique :

```
ln -sv gcc.1 /usr/share/man/man1/cc.1
```

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

```
ln -sfv ../../libexec/gcc/$(gcc -dumpmachine)/15.2.0/liblto_plugin.so \
        /usr/lib/bfd-plugins/
```

Maintenant que notre chaîne d'outils est en place, il est important de s'assurer à nouveau que la compilation et l'édition de liens fonctionneront comme prévu. Vous devez alors effectuer plusieurs contrôles d'intégrité :

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

Il ne devrait pas y avoir d'erreurs et la sortie de la dernière commande aura cette forme (en permettant au nom de l'éditeur des liens de différer en fonction de la plateforme) :

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

Maintenant, assurez-vous que vous êtes prêt à utiliser les bons fichiers :

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

La sortie de la dernière commande devrait être :

```
/usr/lib/gcc/x86_64-pc-linux-gnu/15.2.0/../../../../lib/Scrt1.o succeeded
/usr/lib/gcc/x86_64-pc-linux-gnu/15.2.0/../../../../lib/crti.o succeeded
/usr/lib/gcc/x86_64-pc-linux-gnu/15.2.0/../../../../lib/crtn.o succeeded
```

Selon l'architecture de votre machine, le message ci-dessus peut légèrement différer. La différence porte sur le nom du répertoire après `/usr/lib/gcc`. Il est important de vérifier que **gcc** a trouvé les trois fichiers `crt*.o` sous le répertoire `/usr/lib`.

Vérifiez que le compilateur recherche les bons fichiers d'en-têtes :

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

Cette commande devrait renvoyer la sortie suivante :

```
#include <...> search starts here:
 /usr/lib/gcc/x86_64-pc-linux-gnu/15.2.0/include
 /usr/local/include
 /usr/lib/gcc/x86_64-pc-linux-gnu/15.2.0/include-fixed
 /usr/include
```

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

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

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

Les références au chemins qui ont des composantes comme « -linux-gnu » devraient être ignorés, mais sinon la sortie de la dernière commande devrait être :

```
SEARCH_DIR("/usr/x86_64-pc-linux-gnu/lib64")
SEARCH_DIR("/usr/local/lib64")
SEARCH_DIR("/lib64")
SEARCH_DIR("/usr/lib64")
SEARCH_DIR("/usr/x86_64-pc-linux-gnu/lib")
SEARCH_DIR("/usr/local/lib")
SEARCH_DIR("/lib")
SEARCH_DIR("/usr/lib");
```

Un système 32 bits peut voir quelques répertoires différemment. Par exemple, voici la sortie d'une machine i686 :

```
SEARCH_DIR("/usr/i686-pc-linux-gnu/lib32")
SEARCH_DIR("/usr/local/lib32")
SEARCH_DIR("/lib32")
SEARCH_DIR("/usr/lib32")
SEARCH_DIR("/usr/i686-pc-linux-gnu/lib")
SEARCH_DIR("/usr/local/lib")
SEARCH_DIR("/lib")
SEARCH_DIR("/usr/lib");
```

Ensuite assurez-vous que vous utilisez la bonne libc :

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

La sortie de la dernière commande devrait être :

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

Assurez-vous que GCC utilise le bon éditeur dynamique :

```
grep found dummy.log
```

La sortie de la dernière commande devrait être (avec éventuellement des différences spécifiques à votre plateforme dans le nom de l'éditeur dynamique) :

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

Si la sortie ne ressemble pas à celle montrée ci-dessus ou si elle n'est pas disponible du tout, cela signifie que quelque chose s'est vraiment mal passé. Enquêtez et répétez les étapes pour trouver où les problèmes se trouvent et corrigez-les. Tout problème doit être résolu avant de continuer le processus.

Une fois que tout fonctionne correctement, nettoyez les fichiers de test :

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

Enfin, déplacez un fichier mal placé :

```
mkdir -pv /usr/share/gdb/auto-load/usr/lib
mv -v /usr/lib/*gdb.py /usr/share/gdb/auto-load/usr/lib
```

8.29.2. Contenu de GCC

Programmes installés: c++, cc (lien vers gcc), cpp, g++, gcc, gcc-ar, gcc-nm, gcc-ranlib, gcov, gcov-dump, gcov-tool et lto-dump

Bibliothèques installées: libasan.{a,so}, libatomic.{a,so}, libcc1.so, libgcc.a, libgcc_eh.a, libgcc_s.so, libgcov.a, libgomp.{a,so}, libhwasan.{a,so}, libitm.{a,so}, liblsan.{a,so}, liblto_plugin.so, libquadmath.{a,so}, libssp.{a,so}, libssp_nonshared.a, libstdc++.{a,so}, libstdc++exp.a, libstdc++fs.a, libsupc++.a, libtsan.{a,so} et libubsan.{a,so}

Répertoires installés: /usr/include/c++, /usr/lib/gcc, /usr/libexec/gcc et /usr/share/gcc-15.2.0

Descriptions courtes

c++	Le compilateur C++
cc	Le compilateur C
cpp	Le pré-processeur C est utilisé par le compilateur pour l'extension des instructions #include, #define et d'autres instructions similaires dans les fichiers source
g++	Le compilateur C++
gcc	Le compilateur C
gcc-ar	Une enveloppe autour de **ar** qui ajoute un greffon à la ligne de commande. Ce programme n'est utilisé que pour ajouter « l'optimisation à l'édition des liens » et il n'est pas utile avec les options de construction par défaut.
gcc-nm	Une enveloppe autour de **nm** qui ajoute un greffon à la ligne de commande. Ce programme n'est utilisé que pour ajouter « l'optimisation à l'édition des liens » et il n'est pas utile avec les options de construction par défaut.
gcc-ranlib	Une enveloppe autour de **ranlib** qui ajoute un greffon à la ligne de commande. Ce programme n'est utilisé que pour ajouter « l'optimisation à l'édition des liens » et il n'est pas utile avec les options de construction par défaut.
gcov	Un outil de tests, qui est utilisé pour analyser les programmes et savoir où des optimisations seraient suivies du plus d'effets
gcov-dump	Outil d'affichage de profil gcda et gcno hors-ligne
gcov-tool	Outil de traitement de profils gcda hors-ligne
lto-dump	Outil pour afficher les fichiers objets produits par GCC quand LTO est activé
`libasan`	La bibliothèque de vérification des adresses à l'exécution
`libatomic`	Bibliothèque d'exécution intégrée pour les opérations atomiques de GCC
`libcc1`	Une bibliothèque qui permet à GDB d'utiliser GCC
`libgcc`	Contient la prise en charge de **gcc** à l'exécution
`libgcov`	Cette bibliothèque est liée à un programme si GCC active le profilage
`libgomp`	L'implémentation GNU de l'API OpenMP API pour la programmation en mémoire parallèle partagée sur plusieurs plateformes en C/C++ et Fortran
`libhwasan`	La bibliothèque de vérification des adresses à assistance matérielle à l'exécution
`libitm`	La bibliothèque mémoire transactionnelle de GNU
`liblsan`	La bibliothèque de vérification de fuites à l'exécution
`liblto_plugin`	Le greffon LTO de GCC permet à Binutils de traiter les fichiers objets créés par GCC quand LTO est activé
`libquadmath`	API de la bibliothèque mathématique en quadruple précision de GCC
`libssp`	Contient des routines qui prennent en charge la fonctionnalité de protection de GCC contre les débordement de pile. Normalement elle n'est pas utilisée car glibc fournit aussi ces routines.
`libstdc++`	La bibliothèque C++ standard
`libstdc++exp`	Bibliothèque de contrats C++ expérimentale
`libstdc++fs`	Bibliothèque de systèmes de fichiers ISO/IEC TS 18822:2015
`libsupc++`	Fournit des routines de prise en charge du langage de programmation C++
`libtsan`	La bibliothèque de vérification des threads à l'exécution

`libubsan`	La bibliothèque de vérification des comportements non définis à l'exécution

8.30. Ncurses-6.5-20250809

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

Temps de construction approximatif: 0,2 SBU
Espace disque requis: 46 Mo

8.30.1. Installation de Ncurses

Préparez la compilation de Ncurses :

```
./configure --prefix=/usr           \
            --mandir=/usr/share/man \
            --with-shared           \
            --without-debug         \
            --without-normal        \
            --with-cxx-shared       \
            --enable-pc-files       \
            --with-pkg-config-libdir=/usr/lib/pkgconfig
```

Voici la signification des nouvelles options de configure :

`--with-shared`
Cette option fait construire et installer les bibliothèques C partagées de Ncurses.

`--without-normal`
Cette option empêche Ncurses de construire et d'installer les bibliothèques C statiques.

`--without-debug`
Cette option empêche Ncurses de construire et d'installer les bibliothèques de débogage.

`--with-cxx-shared`
Cela fait construire et installer les liaisons C++ partagées de Ncurses. Cela l'empêche également de construire et d'installer les liaisons C++ statiques.

`--enable-pc-files`
Ce paramètre génère et installe les fichiers .pc pour pkg-config.

Compilez le paquet :

```
make
```

Ce paquet a une suite de tests, mais elle ne peut être exécutée qu'après l'installation du paquet. Les tests se situent dans le répertoire `test/`. Voir le fichier `README` dans ce répertoire pour de plus amples détails.

L'installation de ce paquet écrasera `libncursesw.so.6.5`. Cela peut faire crasher le processus de shell qui utilise du code et des données du fichier de bibliothèque. Installez le paquet avec `DESTDIR`, et remplacez le fichier de bibliothèque correctement avec la commande **install** (l'en-tête `curses.h` est également modifié pour s'assurer que l'ABI des caractères larges est utilisée comme nous l'avons fait dans Section 6.3, « Ncurses-6.5-20250809 ») :

```
make DESTDIR=$PWD/dest install
install -vm755 dest/usr/lib/libncursesw.so.6.5 /usr/lib
rm -v  dest/usr/lib/libncursesw.so.6.5
sed -e 's/^#if.*XOPEN.*$/#if 1/' \
    -i dest/usr/include/curses.h
cp -av dest/* /
```

Beaucoup d'applications s'attendent encore à ce que l'éditeur de liens puisse trouver les bibliothèques Ncurses non wide-character. Faites en sorte que ces applications croient au lien vers les bibliothèques wide-character par des liens symboliques (remarquez que les liens `.so` ne sont surs que pour `curses.h` modifié pour toujours utiliser l'ABI wide-character) :

```
for lib in ncurses form panel menu ; do
    ln -sfv lib${lib}w.so /usr/lib/lib${lib}.so
    ln -sfv ${lib}w.pc    /usr/lib/pkgconfig/${lib}.pc
done
```

Finalement, assurez-vous que les vieilles applications qui cherchent `-lcurses` lors de la compilation sont encore compilables :

```
ln -sfv libncursesw.so /usr/lib/libcurses.so
```

Si désiré, installez la documentation de Ncurses :

```
cp -v -R doc -T /usr/share/doc/ncurses-6.5-20250809
```

Note

Les instructions ci-dessus ne créent pas de bibliothèques Ncurses non-wide-character puisqu'aucun paquet installé par la compilation à partir des sources ne se lie à elles lors de l'exécution. Pour le moment, les seules applications binaires connues qui se lient aux bibliothèques Ncurses non-wide-character exigent la version 5. Si vous devez avoir de telles bibliothèques à cause d'une application disponible uniquement en binaire ou pour vous conformer à la LSB, compilez à nouveau le paquet avec les commandes suivantes :

```
make distclean
./configure --prefix=/usr    \
            --with-shared    \
            --without-normal \
            --without-debug  \
            --without-cxx-binding \
            --with-abi-version=5
make sources libs
cp -av lib/lib*.so.5* /usr/lib
```

8.30.2. Contenu de Ncurses

Programmes installés: captoinfo (lien vers tic), clear, infocmp, infotocap (lien vers tic), ncursesw6-config, reset (lien vers tset), tabs, tic, toe, tput et tset

Bibliothèques installées: libcurses.so (lien symbolique), libform.so (lien symbolique), libformw.so, libmenu.so (symlink), libmenuw.so, libncurses.so (lien symbolique), libncursesw.so, libncurses++w.so, libpanel.so (lien symbolique) et libpanelw.so,

Répertoires installés: /usr/share/tabset, /usr/share/terminfo et /usr/share/doc/ncurses-6.5-20250809

Descriptions courtes

captoinfo	Convertit une description termcap en description terminfo
clear	Efface l'écran si possible
infocmp	Compare ou affiche les descriptions terminfo
infotocap	Convertit une description terminfo en description termcap
ncursesw6-config	Fournit des informations de configuration de ncurses
reset	Réinitialise un terminal avec ses valeurs par défaut
tabs	Efface et initialise des taquets de tab sur un terminal

tic	Le compilateur d'entrée de description terminfo qui traduit un fichier terminfo au format source dans un format binaire nécessaire pour les routines des bibliothèques ncurses. Un fichier terminfo contient des informations sur les capacités d'un terminal donné
toe	Liste tous les types de terminaux disponibles, donnant pour chacun d'entre eux son nom principal et sa description
tput	Rend les valeurs de capacités dépendant du terminal disponibles au shell ; il peut aussi être utilisé pour réinitialiser un terminal ou pour afficher son nom complet
tset	Peut être utilisé pour initialiser des terminaux
`libncursesw`	Contient des fonctions pour afficher du texte de plusieurs façons complexes sur un écran de terminal ; un bon exemple d'utilisation de ces fonctions est le menu affiché par le **make menuconfig** du noyau
`libncurses++w`	Contient les liaisons C++ pour les autres bibliothèques de ce paquet
`libformw`	Contient des fonctions pour implémenter des formulaires
`libmenuw`	Contient des fonctions pour implémenter des menus
`libpanelw`	Contient des fonctions pour implémenter des panneaux

8.31. Sed-4.9

Le paquet Sed contient un éditeur de flux.

Temps de construction approximatif: 0,3 SBU
Espace disque requis: 30 Mo

8.31.1. Installation de Sed

Préparez la compilation de Sed :

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

Compilez le paquet et générez la documentation HTML :

```
make
make html
```

Pour tester les résultats, exécutez :

```
chown -R tester .
su tester -c "PATH=$PATH make check"
```

Installez le paquet et sa documentation :

```
make install
install -d -m755           /usr/share/doc/sed-4.9
install -m644 doc/sed.html /usr/share/doc/sed-4.9
```

8.31.2. Contenu de Sed

Programme installé: sed
Répertoire installé: /usr/share/doc/sed-4.9

Descriptions courtes

sed Filtre et transforme des fichiers texte en une seule passe

8.32. Psmisc-23.7

Le paquet Psmisc contient des programmes pour afficher des informations sur les processus en cours d'exécution.

Temps de construction approximatif: moins de 0,1 SBU
Espace disque requis: 6,7 Mo

8.32.1. Installation de Psmisc

Préparez la compilation de Psmisc pour :

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

Compilez le paquet :

```
make
```

Pour lancer la suite de tests, exécutez :

```
make check
```

Installez le paquet :

```
make install
```

8.32.2. Contenu de Psmisc

Programmes installés: fuser, killall, peekfd, prtstat, pslog, pstree et pstree.x11 (lien vers pstree)

Descriptions courtes

fuser Indique les PID de processus utilisant les fichiers ou systèmes de fichiers donnés
killall Tue les processus suivant leur nom. Il envoie un signal à tous les processus en cours
peekfd Observe les descripteurs d'un processus en cours d'exécution, selon son PID
prtstat Affiche des informations sur un processus
pslog Rapport le chemin du journal actuel d'un processus
pstree Affiche les processus en cours hiérarchiquement
pstree.x11 Identique à **pstree**, si ce n'est qu'il attend une confirmation avant de quitter

8.33. Gettext-0.26

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

Temps de construction approximatif: 2,1 SBU
Espace disque requis: 395 Mo

8.33.1. Installation de Gettext

Préparez la compilation de Gettext :

```
./configure --prefix=/usr    \
            --disable-static \
            --docdir=/usr/share/doc/gettext-0.26
```

Compilez le paquet :

```
make
```

Pour tester les résultats, exécutez :

```
make check
```

Installez le paquet :

```
make install
chmod -v 0755 /usr/lib/preloadable_libintl.so
```

8.33.2. Contenu de Gettext

Programmes installés: autopoint, envsubst, gettext, gettext.sh, gettextize, msgattrib, msgcat, msgcmp, msgcomm, msgconv, msgen, msgexec, msgfilter, msgfmt, msggrep, msginit, msgmerge, msgunfmt, msguniq, ngettext, recode-sr-latin et xgettext
Bibliothèques installées: libasprintf.so, libgettextlib.so, libgettextpo.so, libgettextsrc.so, libtextstyle.so et preloadable_libintl.so
Répertoires installés: /usr/lib/gettext, /usr/share/doc/gettext-0.26, /usr/share/gettext et /usr/share/gettext-0.26

Descriptions courtes

autopoint Copie les fichiers d'infrastructure standard Gettext en un paquet source
envsubst Substitue les variables d'environnement dans des chaînes de format shell
gettext Traduit un message en langue naturelle dans la langue de l'utilisateur en recherchant la traduction dans un catalogue de messages
gettext.sh Sert en priorité de bibliothèque de fonction shell pour gettext
gettextize Copie tous les fichiers standard Gettext dans le répertoire de haut niveau d'un paquet, pour commencer son internationalisation
msgattrib Filtre les messages d'un catalogue de traduction suivant leurs attributs et manipule les attributs
msgcat Concatène et fusionne les fichiers `.po`
msgcmp Compare deux fichiers `.po` pour vérifier que les deux contiennent le même ensemble de chaînes msgid

msgcomm	Trouve les messages qui sont communs aux fichiers `.po` donnés
msgconv	Convertit un catalogue de traduction en un autre codage de caractères
msgen	Crée un catalogue de traduction anglais
msgexec	Applique une commande pour toutes les traductions d'un catalogue de traduction
msgfilter	Applique un filtre à toutes les traductions d'un catalogue de traductions
msgfmt	Génère un catalogue binaire de messages à partir d'un catalogue de traductions
msggrep	Extrait tous les messages d'un catalogue de traductions correspondant à un modèle donné ou appartenant à d'autres sources données
msginit	Crée un nouveau fichier `.po`, initialise l'environnement de l'utilisateur
msgmerge	Combine deux traductions brutes en un seul fichier
msgunfmt	Décompile un catalogue de messages binaires en un texte brut de la traduction
msguniq	Unifie les traductions dupliquées en un catalogue de traduction
ngettext	Affiche les traductions dans la langue native d'un message texte dont la forme grammaticale dépend d'un nombre
recode-sr-latin	Recode du texte serbe de l'écrit cyrillique au latin
xgettext	Extrait les lignes de messages traduisibles à partir des fichiers source donnés pour réaliser la première traduction de modèle
`libasprintf`	Définit la classe *autosprintf* qui rend les routines de sortie formatée C utilisables dans les programmes C++ pour utiliser les chaînes de <*string*> et les flux de <*iostream*>
`libgettextlib`	Contient les routines communes utilisées par les nombreux programmes Gettext. Ils ne sont pas faits pour un emploi général
`libgettextpo`	Utilisé pour écrire les programmes spécialisés qui s'occupent des fichiers `.po`. Cette bibliothèque est utilisée lorsque les applications standards livrées avec Gettext ne vont pas suffire (comme **msgcomm**, **msgcmp**, **msgattrib** et **msgen**)
`libgettextsrc`	Fournit des routines communes utilisées par les nombreux programmes Gettext. Ils ne sont pas faits pour un emploi général
`libtextstyle`	Bibliothèque de mise en forme de texte
`preloadable_libintl`	Une bibliothèque faite pour être utilisée par LD_PRELOAD et qui aide `libintl` à archiver des messages non traduits

8.34. Bison-3.8.2

Le paquet Bison contient un générateur d'analyseurs.

Temps de construction approximatif: 2,1 SBU
Espace disque requis: 63 Mo

8.34.1. Installation de Bison

Préparez la compilation de Bison :

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

Compilez le paquet :

```
make
```

Pour tester les résultats, exécutez :

```
make check
```

Installez le paquet :

```
make install
```

8.34.2. Contenu de Bison

Programmes installés: bison et yacc
Bibliothèque installée: liby.a
Répertoire installé: /usr/share/bison

Descriptions courtes

bison Génère, à partir d'une série de règles, un programme d'analyse de structure de fichiers texte ; Bison est un remplacement pour Yacc (Yet Another Compiler Compiler)

yacc Un emballage pour **bison**, utile pour les programmes qui appellent toujours **yacc** au lieu de **bison** ; il appelle **bison** avec l'option *-y*

`liby` La bibliothèque Yacc contenant des implémentations, compatible Yacc, des fonctions `yyerror` et `main` ; cette bibliothèque n'est généralement pas très utile mais POSIX en a besoin

8.35. Grep-3.12

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

Temps de construction approximatif: 0,6 SBU
Espace disque requis: 48 Mo

8.35.1. Installation de Grep

Tout d'abord, supprimez un avertissement sur l'utilisation d'egrep et fgrep qui fount échouer les tests de certains paquets :

```
sed -i "s/echo/#echo/" src/egrep.sh
```

Préparez la compilation de Grep :

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

Compilez le paquet :

```
make
```

Pour tester les résultats, exécutez :

```
make check
```

Installez le paquet :

```
make install
```

8.35.2. Contenu de Grep

Programmes installés: egrep, fgrep et grep

Descriptions courtes

egrep Affiche les lignes qui correspondent à une expression rationnelle étendue. Il est obsolète, utilisez plutôt **grep -E**

fgrep Affiche les lignes qui correspondent à une liste de chaînes fixes. Il est obsolète, utilisez plutôt **grep -F**

grep Affiche des lignes qui correspondent à une expression rationnelle basique

8.36. Bash-5.3

Le paquet Bash contient le Bourne-Again Shell.

Temps de construction approximatif: 1,5 SBU
Espace disque requis: 56 Mo

8.36.1. Installation de Bash

Préparez la compilation de Bash :

```
./configure --prefix=/usr             \
            --without-bash-malloc     \
            --with-installed-readline \
            --docdir=/usr/share/doc/bash-5.3
```

Voici la signification de la nouvelle option de configuration :

`--with-installed-readline`
Cette option indique à Bash d'utiliser la bibliothèque `readline` déjà installée sur le système plutôt que d'utiliser sa propre version de readline.

Compilez le paquet :

```
make
```

Passez à la partie « Installez le paquet » si vous n'exécutez pas la suite de test.

Pour préparer les tests, assurez-vous que l'utilisateur `tester` peut écrire dans l'arborescence des sources :

```
chown -R tester .
```

La suite de tests de ce paquet est conçue pour être lancée en tant qu'utilisateur non-`root` qui possède le terminal connecté à l'entrée standard. Pour satisfaire ce prérequis, démarrez un nouveau pseudo-terminal avec Expect et lancez les tests en tant qu'utilisateur `tester` :

```
LC_ALL=C.UTF-8 su -s /usr/bin/expect tester << "EOF"
set timeout -1
spawn make tests
expect eof
lassign [wait] _ _ _ value
exit $value
EOF
```

La suite de tests utilise **diff** pour détecter les différences entre la sortie des scripts de test et la sortie attendue. Toute sortie de **diff** (préfixée par `<` et `>`) indique un échec du test, à moins qu'un message disant que la différence est ignorée n'apparaisse. Le test nommé `run-builtins` est connu pour échouer sur certaines distributions hôtes avec une différence sur les lignes 479 et 480 de la sortie. Certains autres tests ont besoin des paramètres linguistiques `zh_TW.BIG5` et `ja_JP.SJIS` et sont connus pour échouer s'ils ne sont pas installés.

Installez le paquet :

```
make install
```

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

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

8.36.2. Contenu de Bash

Programmes installés: bash, bashbug et sh (lien vers bash)
Répertoire installé: /usr/include/bash, /usr/lib/bash et /usr/share/doc/bash-5.3

Descriptions courtes

bash Un interpréteur de commandes largement utilisé ; il réalise un grand nombre d'expansions et de substitutions sur une ligne de commande donnée avant de l'exécuter, ce qui fait de cet interpréteur un outil très puissant

bashbug Un script shell qui aide l'utilisateur à composer et à envoyer des courriers électroniques contenant des rapports de bogues formatés concernant **bash**

sh Un lien symbolique vers le programme **bash** ; à son appel en tant que **sh**, **bash** essaie de copier le comportement initial des versions historiques de **sh** aussi fidèlement que possible, tout en se conformant aussi au standard POSIX

8.37. Libtool-2.5.4

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

Temps de construction approximatif: 0,6 SBU
Espace disque requis: 44 Mo

8.37.1. Installation de Libtool

Préparez la compilation de Libtool :

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

Compilez le paquet :

```
make
```

Pour tester les résultats, exécutez :

```
make check
```

Installez le paquet :

```
make install
```

Supprimez une bibliothèque statique qui n'est utile que pour la suite de tests :

```
rm -fv /usr/lib/libltdl.a
```

8.37.2. Contenu de Libtool

Programmes installés: libtool et libtoolize
Bibliothèques installées: libltdl.so
Répertoires installés: /usr/include/libltdl, et /usr/share/libtool

Descriptions courtes

libtool Fournit des services de prise en charge de construction généralisée de bibliothèques

libtoolize Fournit une façon standard d'ajouter la prise en charge de **libtool** dans un paquet

`libltdl` Cache les difficultés d'ouverture des bibliothèques dynamiques chargées

8.38. GDBM-1.26

Le paquet GDBM contient le gestionnaire de bases de données de GNU. C'est une bibliothèque de fonctions de bases de données qui utilise du hachage extensible et qui fonctionne comme le dbm standard d'UNIX. La bibliothèque offre les bases pour stocker des paires clés/données, chercher et extraire les données avec leur clé, effacer celles-ci ainsi que leurs données associées.

Temps de construction approximatif: moins de 0,2 SBU
Espace disque requis: 13 Mo

8.38.1. Installation de GDBM

Préparez la compilation de GDBM :

```
./configure --prefix=/usr    \
            --disable-static \
            --enable-libgdbm-compat
```

Voici la signification de l'option de configuration :

`--enable-libgdbm-compat`
Ce paramètre permet de construire la bibliothèque de compatibilité libgdbm. D'autres paquets extérieurs à LFS peuvent exiger les anciennes routines de DBM qu'elle fournit.

Compilez le paquet :

```
make
```

Pour tester les résultats, exécutez :

```
make check
```

Installez le paquet :

```
make install
```

8.38.2. Contenu de GDBM

Programmes installés: gdbm_dump, gdbm_load, et gdbmtool
Bibliothèques installées: libgdbm.so et libgdbm_compat.so

Descriptions courtes

gdbm_dump Envoie une base de données GDBM vers un fichier
gdbm_load Recrée une base de données GDBM à partir d'un fichier
gdbmtool Règle et modifie une base de données GDBM
`libgdbm` Contient des fonctions pour manipuler une base de données hachée
`libgdbm_compat` Bibliothèque de compatibilité contenant les anciennes fonctions DBM

8.39. Gperf-3.3

Gperf génère une fonction de hachage parfait à partir d'un trousseau.

Temps de construction approximatif: 0,2 SBU
Espace disque requis: 12 Mo

8.39.1. Installation de Gperf

Préparez la compilation de Gperf :

```
./configure --prefix=/usr --docdir=/usr/share/doc/gperf-3.3
```

Compilez le paquet :

```
make
```

Pour tester les résultats, exécutez :

```
make check
```

Installez le paquet :

```
make install
```

8.39.2. Contenu de Gperf

Programme installé: gperf
Répertoire installé: /usr/share/doc/gperf-3.3

Descriptions courtes

gperf Génère un hachage parfait à partir d'un trousseau

8.40. Expat-2.7.1

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

Temps de construction approximatif: 0,1 SBU
Espace disque requis: 14 Mo

8.40.1. Installation d'Expat

Préparez la compilation d'Expat :

```
./configure --prefix=/usr    \
            --disable-static \
            --docdir=/usr/share/doc/expat-2.7.1
```

Compilez le paquet :

```
make
```

Pour tester les résultats, exécutez :

```
make check
```

Installez le paquet :

```
make install
```

Si vous le souhaitez, installez la documentation :

```
install -v -m644 doc/*.{html,css} /usr/share/doc/expat-2.7.1
```

8.40.2. Contenu d'Expat

Programme installé: xmlwf
Bibliothèques installées: libexpat.so
Répertoire installé: /usr/share/doc/expat-2.7.1

Descriptions courtes

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

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

8.41. Inetutils-2.6

Le paquet Inetutils contient des programmes réseaux basiques.

Temps de construction approximatif: 0,3 SBU
Espace disque requis: 36 Mo

8.41.1. Installation de Inetutils

Tout d'abord, assurez-vous que le paquet construise avec gcc-14.1 ou supérieur :

```
sed -i 's/def HAVE_TERMCAP_TGETENT/ 1/' telnet/telnet.c
```

Préparez la compilation d'Inetutils :

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

Voici la signification des options de configuration :

`--disable-logger`
Cette option empêche l'installation du programme **logger** par Inetutils. Ce programme est utilisé par les scripts pour passer des messages au démon des traces système. Nous ne l'installons pas car Util-linux livre une version plus récente.

`--disable-whois`
Cette option désactive la construction du client **whois** d'Inetutils qui est vraiment obsolète. Les instructions pour un meilleur client **whois** sont dans le livre BLFS.

`--disable-r*`
Ces paramètres désactivent la construction de programmes obsolètes qui ne doivent pas être utilisés pour des raisons de sécurité. Les fonctions fournies pas ces programmes peuvent être fournies par le paquet openssh du livre BLFS.

`--disable-servers`
Ceci désactive l'installation des différents serveurs réseau inclus dans le paquet Inetutils. Ces serveurs semblent inappropriés dans un système LFS de base. Certains ne sont pas sécurisés et ne sont considérés sûrs que sur des réseaux de confiance. Remarquez que de meilleurs remplacements sont disponibles pour certains de ces serveurs.

Compilez le paquet :

```
make
```

Pour tester les résultats, exécutez :

```
make check
```

Un test nommé `libls.sh` est connu pour parfois échouer.

Installez le paquet :

```
make install
```

Déplacez un programme au bon emplacement :

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

8.41.2. Contenu de Inetutils

Programmes installés: dnsdomainname, ftp, ifconfig, hostname, ping, ping6, talk, telnet, tftp et traceroute

Descriptions courtes

dnsdomainname	Affiche le nom de domaine du système
ftp	Est le programme de transfert de fichier
hostname	Affiche ou règle le nom de l'hôte
ifconfig	Gère des interfaces réseaux
ping	Envoie des paquets echo-request et affiche le temps mis pour que la réponse arrive
ping6	Une version de **ping** pour les réseaux IPv6
talk	Est utilisé pour discuter avec un autre utilisateur
telnet	Une interface du protocole TELNET
tftp	Un programme de transfert trivial de fichiers
traceroute	Trace le trajet que prennent vos paquets depuis l'endroit où vous travaillez jusqu'à un hôte sur un réseau, en montrant tous les hops (passerelles) intermédiaires pendant le chemin

8.42. Less-679

Le paquet Less contient un visualiseur de fichiers texte.

Temps de construction approximatif: moins de 0,1 SBU
Espace disque requis: 16 Mo

8.42.1. Installation de Less

Préparez la compilation de Less :

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

Voici la signification des options de configuration :

`--sysconfdir=/etc`
Cette option indique aux programmes créés par le paquet de chercher leurs fichiers de configuration dans `/etc`.

Compilez le paquet :

```
make
```

Pour tester les résultats, exécutez :

```
make check
```

Installez le paquet :

```
make install
```

8.42.2. Contenu de Less

Programmes installés: less, lessecho et lesskey

Descriptions courtes

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

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

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

8.43. Perl-5.42.0

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

Temps de construction approximatif: 1,3 SBU
Espace disque requis: 257 Mo

8.43.1. Installation de Perl

Cette version de Perl compile à présent les modules Compress::Raw::Zlib et Compress::Raw::BZip2. Par défaut, Perl utilisera une copie interne du code source Zlib pour la compilation. Lancez la commande suivante afin que Perl utilise les bibliothèques Zlib installées sur le système :

```
export BUILD_ZLIB=False
export BUILD_BZIP2=0
```

Si vous voulez avoir un contrôle total sur la façon dont Perl est configuré, vous pouvez supprimer les options « -des » de la commande suivante et contrôler à la main la façon dont ce paquet est construit. Autrement, exécutez l'exacte commande ci-dessous pour utiliser les paramètres par défaut que détecte Perl automatiquement :

```
sh Configure -des                                         \
             -D prefix=/usr                               \
             -D vendorprefix=/usr                         \
             -D privlib=/usr/lib/perl5/5.42/core_perl     \
             -D archlib=/usr/lib/perl5/5.42/core_perl     \
             -D sitelib=/usr/lib/perl5/5.42/site_perl     \
             -D sitearch=/usr/lib/perl5/5.42/site_perl    \
             -D vendorlib=/usr/lib/perl5/5.42/vendor_perl  \
             -D vendorarch=/usr/lib/perl5/5.42/vendor_perl \
             -D man1dir=/usr/share/man/man1               \
             -D man3dir=/usr/share/man/man3               \
             -D pager="/usr/bin/less -isR"                \
             -D useshrplib                                \
             -D usethreads
```

Voici la signification des nouvelles options de configuration :

`-D pager="/usr/bin/less -isR"`
Ceci assure que **`less`** est utilisé au lieu de **`more`**.

`-D man1dir=/usr/share/man/man1 -D man3dir=/usr/share/man/man3`
Étant donné que Groff n'est pas encore installé, **Configure** pense que nous ne voulons pas des pages de manuel de Perl. Ces paramètres changent cette décision.

`-D usethreads`
Construisez perl avec la prise en charge des threads.

Compilez le paquet :

```
make
```

Pour tester les résultats, exécutez :

```
TEST_JOBS=$(nproc) make test_harness
```

Installez le paquet et faites le ménage :

```
make install
unset BUILD_ZLIB BUILD_BZIP2
```

8.43.2. Contenu de Perl

Programmes installés: corelist, cpan, enc2xs, encguess, h2ph, h2xs, instmodsh, json_pp, libnetcfg, perl, perl5.42.0 (lien matériel vers perl), perlbug, perldoc, perlivp, perlthanks (lien matériel vers perlbug), piconv, pl2pm, pod2html, pod2man, pod2text, pod2usage, podchecker, podselect, prove, ptar, ptardiff, ptargrep, shasum, splain, xsubpp et zipdetails
Bibliothèques installées: Plusieurs qui ne peuvent pas être listés ici
Répertoire installé: /usr/lib/perl5

Descriptions courtes

corelist Une interface en ligne de commande pour Module::CoreList

cpan Interagit avec le réseau d'archive Perl global (*Comprehensive Perl Archive Network*, CPAN) à partir de la ligne de commande

enc2xs Construit une extension Perl pour le module Encode, soit à partir de *Unicode Character Mappings* soit à partir de *Tcl Encoding Files*

encguess Devine le type d'encodage d'un ou plusieurs fichiers

h2ph Convertit les fichiers d'en-têtes C `.h` en fichiers d'en-têtes Perl `.ph`

h2xs Convertit les fichiers d'en-têtes C `.h` en extensions Perl

instmodsh Script shell pour examiner les modules Perl installés, et pouvant créer une archive tar à partir d'un module installé

json_pp Convertit des données entre certains formats d'entrée et de sortie

libnetcfg Peut être utilisé pour configurer le module Perl `libnet`

perl Combine quelques-unes des meilleures fonctionnalités de C, **sed**, **awk** et **sh** en un langage style couteau suisse

perl5.42.0 Un lien matériel vers **perl**

perlbug Utilisé pour générer des rapports de bogues sur Perl ou les modules l'accompagnant et pour les envoyer par courrier électronique

perldoc Affiche une partie de la documentation au format pod, embarquée dans le répertoire d'installation de Perl ou dans un script Perl

perlivp La procédure de vérification d'installation de Perl. Il peut être utilisé pour vérifier que Perl et ses bibliothèques ont été installés correctement

perlthanks Utilisé pour générer des messages de remerciements par mail aux développeurs de Perl

piconv Une version Perl du convertisseur de codage des caractères **iconv**

pl2pm Un outil simple pour la conversion des fichiers Perl4 `.pl` en modules Perl5 `.pm`

pod2html Convertit des fichiers à partir du format pod vers le format HTML

pod2man Convertit des fichiers à partir du format pod vers une entrée formatée *roff

pod2text Convertit des fichiers à partir du format pod vers du texte ANSI

pod2usage Affiche les messages d'usage à partir des documents embarqués pod

podchecker Vérifie la syntaxe du format pod des fichiers de documentation

podselect Affiche les sections sélectionnées de la documentation pod

prove Outil en ligne de commande pour lancer des tests liés au module Test::Harness

ptar Un programme du genre **tar** écrit en Perl

ptardiff Un programme Perl qui compare une archive extraite et une non extraite

ptargrep	Un programme Perl qui applique des modèles correspondant au contenu des fichiers d'une archive tar
shasum	Affiche ou vérifie des sommes de contrôle SHA
splain	Utilisé pour forcer la verbosité des messages d'avertissement avec Perl
xsubpp	Convertit le code Perl XS en code C
zipdetails	Affiche des détails sur la structure interne d'un fichier Zip

8.44. XML::Parser-2.47

Le module XML::Parser est une interface Perl avec l'analyseur Expat de James Clark.

Temps de construction approximatif: moins de 0,1 SBU
Espace disque requis: 2,4 Mo

8.44.1. Installation de XML::Parser

Préparez la compilation de XML::Parser :

```
perl Makefile.PL
```

Compilez le paquet :

```
make
```

Pour tester les résultats, exécutez :

```
make test
```

Installez le paquet :

```
make install
```

8.44.2. Contenu de XML::Parser

Module installé: Expat.so

Descriptions courtes

`Expat` fournit l'interface Perl avec Expat

8.45. Intltool-0.51.0

Le paquet Intltool est un outil d'internationalisation utilisé pour extraire des chaînes traduisibles à partir de fichiers sources.

Temps de construction approximatif: moins de 0,1 SBU
Espace disque requis: 1,5 Mo

8.45.1. Installation d'Intltool

Corrigez un avertissement causé par perl-5.22 et les versions ultérieures :

```
sed -i 's:\\\${:\\\$\\{:' intltool-update.in
```

Note

L'expression régulière ci-dessus a l'air inhabituelle à cause des antislashs. Elle ajoute un antislash avant l'accolade ouvrante dans la séquence « \${ » ce qui donne « \&\{ ».

Préparez la compilation d'Intltool :

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

Compilez le paquet :

```
make
```

Pour tester les résultats, exécutez :

```
make check
```

Installez le paquet :

```
make install
install -v -Dm644 doc/I18N-HOWTO /usr/share/doc/intltool-0.51.0/I18N-HOWTO
```

8.45.2. Contenu d'Intltool

Programmes installés: intltool-extract, intltool-merge, intltool-prepare, intltool-update et intltoolize
Répertoires installés: /usr/share/doc/intltool-0.51.0 et /usr/share/intltool

Descriptions courtes

intltoolize Prépare l'utilisation d'intltool par un paquet
intltool-extract Génère des fichiers d'en-tête lisibles par **gettext**
intltool-merge Rassemble les chaînes traduites dans divers types de fichiers
intltool-prepare Met à jour les fichiers pot et les synchronise avec les fichiers de traduction
intltool-update Met à jour les modèles po et les synchronise avec les traductions

8.46. Autoconf-2.72

Le paquet Autoconf contient des programmes conçus pour produire des scripts shell qui configurent automatiquement du code source.

Temps de construction approximatif: moins de 0,1 SBU (environ 0,4 SBU avec les tests)
Espace disque requis: 25 Mo

8.46.1. Installation d’Autoconf

Préparez la compilation d'Autoconf :

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

Compilez le paquet :

```
make
```

Pour tester les résultats, exécutez :

```
make check
```

Installez le paquet :

```
make install
```

8.46.2. Contenu d’Autoconf

Programmes installés: autoconf, autoheader, autom4te, autoreconf, autoscan, autoupdate et ifnames
Répertoire installé: /usr/share/autoconf

Descriptions courtes

autoconf Produit des scripts shell qui configurent automatiquement des paquets de code source logiciel, permettant ainsi de les adapter à tous les types de systèmes Unix. Les scripts de configuration qu'autoconf produit sont indépendants. Il n’y a pas besoin du programme **autoconf** pour les exécuter

autoheader Un outil pour créer des fichiers modèles d'instructions C *#define* que la commande configure pourra utiliser

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 les fichiers modèles d'**autoconf** et d'**automake** sont modifiés

autoscan Aide à la création de fichiers `configure.in` pour un paquet logiciel. Il examine les fichiers source d'une arborescence de répertoires pour y trouver d’éventuels problèmes de portabilité communs, et crée un fichier `configure.scan` servant de fichier `configure.in` préliminaire pour le paquet

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

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

8.47. Automake-1.18.1

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

Temps de construction approximatif: moins de 0,1 SBU (environ 1,1 SBU avec les tests)
Espace disque requis: 123 Mo

8.47.1. Installation de Automake

Préparez la compilation d'Automake :

```
./configure --prefix=/usr --docdir=/usr/share/doc/automake-1.18.1
```

Compilez le paquet :

```
make
```

Utiliser quatre tâches en parallèle accélère la vitesse des tests, même sur les processeurs à moins de cœurs logiques, en raison de délais internes de chaque test. Pour tester les résultats, lancez :

```
make -j$(($(nproc)>4?$(nproc):4)) check
```

Remplacez *`$((...))`* par le nombre de cœurs logiques que vous voulez utiliser si vous ne voulez pas tous les utiliser.

Installez le paquet :

```
make install
```

8.47.2. Contenu de Automake

Programmes installés: aclocal, aclocal-1.18 (lié en dur avec aclocal), automake, et automake-1.18 (lié en dur avec automake)
Répertoires installés: /usr/share/aclocal-1.18, /usr/share/automake-1.18, et /usr/share/doc/automake-1.18.1

Descriptions courtes

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

aclocal-1.18 Un lien matériel vers **aclocal**

automake Un outil pour générer automatiquement des fichiers `Makefile.in` à partir de fichiers `Makefile.am`. Pour créer tous les fichiers `Makefile.in` d'un paquet, lancez ce programme dans le répertoire de haut niveau. En parcourant le fichier `configure.in`, il trouve automatiquement chaque fichier `Makefile.am` approprié et génère le fichier `Makefile.in` correspondant.

automake-1.18 Un lien matériel vers **automake**

8.48. OpenSSL-3.5.2

Le paquet OpenSSL contient des outils de gestion et des bibliothèques cryptographiques. Ils servent à fournir des fonctions cryptographiques à d'autres paquets, comme OpenSSH, des applications de messagerie électronique et des navigateurs Internet (pour accéder à des sites HTTPS).

Temps de construction approximatif: 1,9 SBU
Espace disque requis: 1,1 Go

8.48.1. Installation d'OpenSSL

Préparez la compilation d'OpenSSL :

```
./config --prefix=/usr         \
         --openssldir=/etc/ssl \
         --libdir=lib          \
         shared                \
         zlib-dynamic
```

Compilez le paquet :

```
make
```

Pour tester les résultats, exécutez :

```
HARNESS_JOBS=$(nproc) make test
```

Un test, 30-test_afalg.t, est connu pour échouer si le noyau hôte n'a pas activé `CONFIG_CRYPTO_USER_API_SKCIPHER` ou n'a aucune des options qui fournissent un AES avec l'implémentation CBC (par exemple la combinaison de `CONFIG_CRYPTO_AES` et `CONFIG_CRYPTO_CBC` ou `CONFIG_CRYPTO_AES_NI_INTEL` si le CPU prend AES-NI en charge). S'il échoue, il peut être ignoré sans problème.

Installez le paquet :

```
sed -i '/INSTALL_LIBS/s/libcrypto.a libssl.a//' Makefile
make MANSUFFIX=ssl install
```

Ajoutez la version au nom de répertoire de la documentation, pour rester cohérent avec d'autres paquets :

```
mv -v /usr/share/doc/openssl /usr/share/doc/openssl-3.5.2
```

Si vous le souhaitez, installez de la documentation supplémentainre :

```
cp -vfr doc/* /usr/share/doc/openssl-3.5.2
```

Note

Open SSL doit être mis à jour lorsqu'une nouvelle version corrigeant des vulnérabilités est annoncée. Depuis OpenSSL 3.0.0, les versions d'OpenSSL suivent le schéma MAJEUR.MINEUR.PATCH. La compatibilité d'API/ABI est assurée pour les mêmes numéros de version MAJOR. Comme LFS n'installe que les bibliothèques partagées, vous n'avez pas besoin de recompiler les programmes qui renvoient vers `libcrypto.so` ni `libssl.so` *lorsque vous mettez à jour vers une version qui a le même numéro MAJOR.*

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

8.48.2. Contenu d'OpenSSL

Programmes installés: c_rehash et openssl
Bibliothèques installées: libcrypto.so et libssl.so
Répertoires installés: /etc/ssl, /usr/include/openssl, /usr/lib/engines et /usr/share/doc/openssl-3.5.2

Descriptions courtes

c_rehash est un script Perl qui scanne tous les fichiers dans un répertoire et ajoute des liens symboliques vers leur valeur hashée. L'utilisation de **c_rehash** est considérée comme obsolète et devrait être remplacée par la commande **openssl rehash**

openssl est un outil en ligne de commande qui permet d'utiliser les diverses fonctions cryptographiques de la bibliothèque crypto d'OpenSSL depuis le shell. Il peut être utilisé pour diverses fonctions documentées dans *openssl(1)*

`libcrypto.so` implémente un large éventail d'algorithmes cryptographiques utilisés dans divers standards Internet. Les services fournis par cette bibliothèque sont utilisés par les implémentations OpenSSL de SSL, TLS et S/MIME. Ils ont aussi été utilisés pour implémenter OpenSSH, OpenPGP et d'autres standards de cryptographie

`libssl.so` implémente le protocole *Transport Layer Security* (TLS v1). Elle fournit une API riche, et sa documentation se trouve dans *ssl(7)*

8.49. Libelf de Elfutils-3.12

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

Temps de construction approximatif: 0,3 SBU
Espace disque requis: 156 Mo

8.49.1. Installation de Libelf

Libelf fait partie du paquet elfutils-0.193. Utilisez elfutils-0.193.tar.bz2 comme archive des sources.

Préparez la compilation de Libelf :

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

Compilez le paquet :

```
make
```

Pour tester les résultats, exécutez :

```
make check
```

Deux tests sont connus pour échouer, dwarf_srclang_check et run-backtrace-native-core.sh.

Installez uniquement Libelf :

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

8.49.2. Contenu de Libelf

Bibliothèque installée: libelf.so
Répertoire installé: /usr/include/elfutils

Descriptions courtes

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

8.50. Libffi-3.5.2

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

FFI signifie Foreign Function Interface (Interface de fonction étrangère). Une FFI permet à un programme écrit dans une langue de faire appel à un programme écrit dans une autre langue. Libffi peut notamment créer un pont entre un interprète comme Perl, ou Python, et partager une librairie de sous-programmes écrite en C ou C++.

Temps de construction approximatif: 1,7 SBU
Espace disque requis: 10 Mo

8.50.1. Installation de Libffi

Note

Comme GMP, libffi est construite avec des optimisations spécifiques au processeur utilisé. Si vous construisez pour un autre système, modifiez la valeur du paramètre *--with-gcc-arch=* dans la commande suivante pour spécifier le nom de l'architecture implémentée **à la fois** par le CPU hôte et par le CPU de ce système. Dans le cas contraire, toutes les applications qui se lient à `libffi` afficheront des erreurs de type opération illégale. Si vous n'arrivez pas à trouver une bonne valeur pour les deux CPU, remplacez le paramètre par `--without-gcc-arch` pour produire une bibliothèque générique.

Préparez la compilation de libffi :

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

Voici la signification de l'option de configuration :

`--with-gcc-arch=native`
Vérifie que GCC active les optimisations pour le système actuel. Si l'option n'est pas spécifiée, il essaiera de deviner le système et le code généré peut ne pas être correct pour certains systèmes. Si le code généré est copié du système actuel vers un système avec moins de fonctionnalités, utilisez ce dernier dans le paramètre. Pour des détails concernant les types de systèmes alternatifs, voyez *les options x86 dans le manuel de GCC*.

Compilez le paquet :

```
make
```

Pour tester les résultats, exécutez :

```
make check
```

Installez le paquet :

```
make install
```

8.50.2. Contenu de Libffi

Bibliothèque installée: libffi.so

Descriptions courtes

`libffi` Contient les fonctions de l'API de l'interface pour les fonctions externes

8.51. Python-3.13.7

Le paquet Python 3 contient l'environnement de développement Python. Il est utile pour la programmation orientée objet, écrire des scripts, prototyper de plus grands programmes ou pour développer des applications complètes.

Temps de construction approximatif: 2,0 SBU
Espace disque requis: 453 Mo

8.51.1. Installation de Python 3

Préparez la compilation de Python :

```
./configure --prefix=/usr          \
            --enable-shared        \
            --with-system-expat    \
            --enable-optimizations \
            --without-static-libpython
```

Voici la signification des options de configuration :

`--with-system-expat`
Ce paramètre active la liaison avec la version du système de Expat.

`--enable-optimizations`
Ce paramètre permet d'établir des étapes d'optimisation approfondis, mais prenant beaucoup de temps. L'interprète est construit deux fois ;les tests effectués pendant la première construction sont utilisés pour concevoir la version finale optimisée.

Compilez le paquet :

```
make
```

Certains test sont connus pour ne jamais s'arrêter. Pour tester les résultats, exécutez la suite de tests mais indiquez une limite de 2 minutes pour chaque cas de test :

```
make test TESTOPTS="--timeout 120"
```

Pour un système relativement lent vous devriez augmenter la limite de temps. 1 SBU (mesuré lors de la construction de Binutils passe 1 avec un cœur de CPU) devrait suffire. Certains tests sont instables, donc la suite de tests relancera automatiquement les tests en échec. Si un test échoue mais réussi après une relance, il devrait être considéré comme réussi. Un test, test_ssl, est connu pour échouer dans l'environnement chroot.

Installez le paquet :

```
make install
```

À plusieurs reprises dans ce livre, on utilise la commande **pip3** pour installer les programmes et les modules Python 3 pour chaque utilisateur en tant que `root`. Cela entre en conflit avec les recommandations des développeurs de Python : pour installer des paquets dans un environnement virtuel, ou dans le répertoire home d'un utilisateur régulier (en exécutant **pip3** en tant que cet utilisateur). Un avertissement multi-lignes s'affiche à chaque fois que **pip3** est détecté par l'utilisateur `root`.

La principale raison de cette recommandation est d'éviter les conflits avec le paquet de gestion du système (**dpkg**, par exemple). Ce n'est pas un problème, puisque LFS n'a pas de paquet de gestion s'appliquant à l'intégralité du système. **pip3** vérifiera également si une nouvelle version est disponible dès qu'il sera exécuté. Étant donné que la résolution du nom de domaine n'est pas encore configuré dans l'environnement chroot de LFS, **pip3** ne pourra pas procéder à la recherche d'une mise à jour, et un message d'avertissement s'affichera.

Après avoir lancé le système LFS et établi une connexion au réseau, un nouvel avertissement apparaîtra, prévenant l'utilisateur de mettre à jour **pip3** d'après une roue de pré-construction su PyPI (dès qu'une nouvelle version est disponible). LFS considère cependant **pip3** comme faisant partie de Python 3, il est donc conseillé de ne pas le mettre à jour indépendamment. Une amélioration depuis une roue pré-construite pourrait aussi nous faire dévier de l'objectif : construire un système Linux sur un code source. L'avertissement à propos d'une nouvelle version de **pip3** peut donc lui aussi être ignoré. Vous pouvez, si vous le désirez, désactiver tous ces avertissements en activant la commande suivante, ce qui créera un fichier de configuration :

```
cat > /etc/pip.conf << EOF
[global]
root-user-action = ignore
disable-pip-version-check = true
EOF
```

Important

Dans LFS et BLFS nous construisons et installons normalement les modules Python avec la commande **pip3**. Remarquez bien que les commandes **pip3 install** dans les deux livres doivent être lancées en `root` à moins qu'il s'agisse d'un environnement virtuel Python. Exécuter **pip3 install** en tant qu'utilisateur ou utilisatrice non `root` peut sembler fonctionner, mais cela rendra les modules installés indisponibles pour les autres.

pip3 install ne réinstallera pas les modules déjà installés par défaut. Pour utiliser la commande **pip3 install** pour mettre à jour un module (par exemple, de meson-0.61.3 vers meson-0.62.0), ajoutez l'option *`--upgrade`* à la ligne de commande. S'il est vraiment nécessaire de revenir à une version précédente d'un module ou de réinstaller la même version, ajoutez l'option *`--force-reinstall --no-deps`* à la ligne de commande.

Si vous le souhaitez, installez la documentation préformatée :

```
install -v -dm755 /usr/share/doc/python-3.13.7/html

tar --strip-components=1  \
    --no-same-owner       \
    --no-same-permissions \
    -C /usr/share/doc/python-3.13.7/html \
    -xvf ../python-3.13.7-docs-html.tar.bz2
```

Voici la signification des commandes d'installation de la documentation :

`--no-same-owner` et `--no-same-permissions`
: Garantit que les fichiers installés ont la bonne appartenance et les bonnes permissions. Sans ces options, utiliser tar installera les fichiers du paquet avec les valeurs du créateur en amont.

8.51.2. Contenu de Python 3

Programmes installés: 2to3, idle3, pip3, pydoc3, python3 et python3-config
Bibliothèque installée: libpython3.13.so et libpython3.so
Répertoires installés: /usr/include/python3.13, /usr/lib/python3 et /usr/share/doc/python-3.13.7

Descriptions courtes

2to3 est un programme Python qui lit du code source Python 2.x et applique une série de corrections pour le transformer en code Python 3.x valide

idle3 est un script enveloppe qui ouvre un éditeur en GUI qui connait Python. Pour que ce script puisse tourner, vous devez avoir installé Tk avant Python pour que le module python Tkinter soit construit.

pip3 L'installateur de paquets pour Python. Vous pouvez utiliser pip pour installer des paquets de Python Package Index et d'autres répertoires.

pydoc3 est l'outil de documentation de Python

python3 est un langage de programmation interprété, interactif et orienté objet

8.52. Flit-Core-3.12.0

Flit-core contient les parties de construction de distributions de Flit (un outil de gestion de paquets pour les modules Python simples).

Temps de construction approximatif: moins de 0,1 SBU
Espace disque requis: 1,3 Mo

8.52.1. Installation de Flit-Core

Construisez le paquet :

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

Installez le paquet :

```
pip3 install --no-index --find-links dist flit_core
```

Voici la signification des option de configuration de pip3 et des commandes :

wheel
Cette commande construit l'archive wheel de ce paquet.

`-w dist`
Dit à pip de mettre le wheel créé dans le répertoire `dist`.

`--no-cache-dir`
Évite que pip ne copie le wheel créé vers le répertoire `/root/.cache/pip`.

install
Cette commande installe le paquet.

`--no-build-isolation`, *`--no-deps`* et *`--no-index`*
Ces options empêchent de récupérer des fichiers du répertoire de paquets en ligne (PyPI). Si les paquets sont installés dans le bon ordre, pip ne cherchera aucun fichier dès le départ. Ces options ajoutent un peu de sécurité en cas d'erreur.

`--find-links dist`
Dit à pip de chercher les archives wheel dans le répertoire `dist`.

8.52.2. Contenu de Flit-Core

Répertoire installé: /usr/lib/python3.13/site-packages/flit_core et /usr/lib/python3.13/site-packages/flit_core-3.12.0.dist-info

8.53. Packaging-25.0

Le module packaging est une bibliothèque Python qui fournit des utilitaires qui implémentent les spécifications d'interopérabilités qui ont clairement un seul comportement correct (PEP440) ou bénéficient du fait d'avoir une seule implémentation partagée (PEP425). Cela comprend des utilitaires pour gérer les versions, les spécificateurs, les marqueurs, les étiquettes et les prérequis.

Temps de construction approximatif: moins de 0,1 SBU
Espace disque requis: 3,3 Mo

8.53.1. Installation de Packaging

Compilez packaging avec la commande suivante :

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

Installez packaging avec la commande suivante :

```
pip3 install --no-index --find-links dist packaging
```

8.53.2. Contenu de Packaging

Répertoires installés: /usr/lib/python3.13/site-packages/packaging et /usr/lib/python3.13/site-packages/packaging-25.0.dist-info

8.54. Wheel-0.46.1

Wheel est une bibliothèque Python qui est l'implémentation de référence du standard de gestion des paquets wheel de Python.

Temps de construction approximatif: moins de 0,1 SBU
Espace disque requis: 608 Ko

8.54.1. Installation de Wheel

Compilez wheel avec la commande suivante :

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

Installez wheel avec la commande suivante :

```
pip3 install --no-index --find-links dist wheel
```

8.54.2. Contenu de Wheel

Programme installé: wheel
Répertoires installés: /usr/lib/python3.13/site-packages/wheel et /usr/lib/python3.13/site-packages/wheel-0.37.1-py3.10.egg-info

Descriptions courtes

wheel est un utilitaire pour décompresser, compresser ou convertir des paquets wheel

8.55. Setuptools-80.9.0

Setuptools est un outil utilisé pour télécharger, construire, installer, mettre à jour et désinstaller des paquets Python.

Temps de construction approximatif: moins de 0,1 SBU
Espace disque requis: 25 Mo

8.55.1. Installation de Setuptools

Construisez le paquet :

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

Installez le paquet :

```
pip3 install --no-index --find-links dist setuptools
```

8.55.2. Contenu de Setuptools

Répertoire installé: /usr/lib/python3.13/site-packages/_distutils_hack, /usr/lib/python3.13/site-packages/pkg_resources, /usr/lib/python3.13/site-packages/setuptools et /usr/lib/python3.13/site-packages/setuptools-80.9.0.dist-info

8.56. Ninja-1.13.1

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

Temps de construction approximatif: 0,2 SBU
Espace disque requis: 43 Mo

8.56.1. Installation de Ninja

Lorsqu'il est exécuté, **ninja** lance un nombre maximum de processus en parallèle. Par défaut c'est le nombre de cœurs du système plus deux. Dans certains cas, cela peut surchauffer le CPU ou épuiser la mémoire. Si **ninja** est exécuté depuis la ligne de commande, passer le paramètre -jN limitera le nombre de processus en parallèle, mais certains paquets incluent l'exécution de **ninja** et ne passent pas le paramètre -j.

Utiliser la procédure *facultative* ci-dessous permet à l'utilisateur de limiter le nombre de processus en parallèle via une variable d'environnement, NINJAJOBS. **Par exemple** initialiser :

```
export NINJAJOBS=4
```

limitera **ninja** à 4 processus en parallèle.

Si vous le souhaitez, ajoutez la possibilité à **ninja** d'utiliser la variable d'environnement NINJAJOBS en lançant l'éditeur de flux :

```
sed -i '/int Guess/a \
  int   j = 0;\
  char* jobs = getenv( "NINJAJOBS" );\
  if ( jobs != NULL ) j = atoi( jobs );\
  if ( j > 0 ) return j;\
' src/ninja.cc
```

Construisez Ninja avec :

```
python3 configure.py --bootstrap --verbose
```

Voici la signification des options de construction :

`--bootstrap`
Ce paramètre force ninja à se reconstruire pour le système actuel.

`--verbose`
Ce paramètre fait afficher la progression de la construction de Ninja à **configure.py**.

Les tests du paquet ne peuvent pas être lancés dans l'environnement chroot. Ils nécessitent *cmake*. Cependant, la fonction de base de ce paquet est de toute façon déjà testée en le reconstruisant (avec l'option *`--bootstrap`*).

Installez le paquet :

```
install -vm755 ninja /usr/bin/
install -vDm644 misc/bash-completion /usr/share/bash-completion/completions/ninja
install -vDm644 misc/zsh-completion  /usr/share/zsh/site-functions/_ninja
```

8.56.2. Contenu de Ninja

Programmes installés: ninja

Descriptions courtes

ninja est le système de construction Ninja

8.57. Meson-1.8.3

Meson est un système de construction open source conçu pour être très rapide et aussi convivial que possible.

Temps de construction approximatif: moins de 0,1 SBU
Espace disque requis: 45 Mo

8.57.1. Installation de Meson

Compilez Meson avec la commande suivante :

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

La suite de tests requiert des paquets débordant la portée de LFS.

Installez le paquet :

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

Voici la signification des paramètres d'installation :

`-w dist`
Place les wheels créées dans le répertoire `dist`.

`--find-links dist`
Installe les wheels du répertoire `dist`.

8.57.2. Contenu de Meson

Programmes installés: meson
Répertoire installé: /usr/lib/python3.13/site-packages/meson-1.8.3.dist-info et /usr/lib/python3.13/site-packages/mesonbuild

Descriptions courtes

meson Un système de construction pour une plus grande productivité

8.58. Kmod-34.2

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

Temps de construction approximatif: moins de 0,1 SBU
Espace disque requis: 6,7 Mo

8.58.1. Installation de Kmod

Préparez la compilation de Kmod :

```
mkdir -p build
cd       build

meson setup --prefix=/usr ..    \
            --buildtype=release \
            -D manpages=false
```

Voici la signification des options de configuration :

`-D manpages=false`
Cette option désactive la génération des pages de manuel, car elle nécessite un programme externe.

Compilez le paquet :

```
ninja
```

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

Installez maintenant le paquet :

```
ninja install
```

8.58.2. Contenu de Kmod

Programmes installés: depmod (lien vers kmod), insmod (lien vers kmod), kmod, lsmod (lien vers kmod), modinfo (lien vers kmod), modprobe (lien vers kmod) et rmmod (lien vers kmod)
Bibliothèque installée: libkmod.so

Descriptions courtes

depmod Crée un fichier de dépendances basé sur les symboles qu'il trouve dans l'ensemble de modules existant ; ce fichier de dépendance est utilisé par **modprobe** pour charger automatiquement les modules requis

insmod Installe un module chargeable dans le noyau en cours d'exécution

kmod Charge et décharge les modules du noyau

lsmod Liste les modules actuellement chargés

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

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

rmmod Décharge les modules du noyau en cours d'exécution

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

8.59. Coreutils-9.7

Le paquet Coreutils contient les utilitaires de base requis dans tous les systèmes d'exploitation.

Temps de construction approximatif: 1,2 SBU
Espace disque requis: 180 Mo

8.59.1. Installation de Coreutils

Tout d'abord, appliquez un correctif pour un problème de sécurité identifié en amont :

```
patch -Np1 -i ../coreutils-9.7-upstream_fix-1.patch
```

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

```
patch -Np1 -i ../coreutils-9.7-i18n-1.patch
```

Note

Certains utilisateurs ont trouvé plusieurs bogues dans ce correctif. Avant de signaler des nouveaux bogues aux mainteneurs de Coreutils, veuillez vérifier si les bogues se reproduisent quand le correctif n'est pas déployé.

Maintenant, préparez la compilation de Coreutils :

```
autoreconf -fv
automake -af
FORCE_UNSAFE_CONFIGURE=1 ./configure \
            --prefix=/usr            \
            --enable-no-install-program=kill,uptime
```

Voici la signification des commandes et des options de configuration :

autoreconf -fv
Le correctif pour les paramètres linguistiques a modifié le système de construction, donc les fichiers de configuration doivent être régénérés. Normalement nous aurions utilisé l'option *-i* pour mettre à jour les fichiers auxiliaires standards, mais pour ce paquet cela ne fonctionne pas car `configure.ac` spécifie une ancienne version de gettext.

automake -af
Les fichiers auxiliaires d'automake n'ont pas été mis à jour par **autoreconf** à cause de l'option *-i* manquante. Cette commande les met à jour pour éviter un échec à la construction.

`FORCE_UNSAFE_CONFIGURE=1`
Cette variable d'environnement permet à l'utilisateur `root` de compiler le paquet.

`--enable-no-install-program=kill,uptime`
Ce paramètre empêche Coreutils d'installer des programmes qui seront installés plus tard par d'autres paquets.

Compilez le paquet :

```
make
```

Passez à la partie « Installez le paquet » si vous n'exécutez pas la suite de test.

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

```
make NON_ROOT_USERNAME=tester check-root
```

Nous allons exécuter le reste des tests en tant qu'utilisateur `tester`. Certains tests exigent cependant que l'utilisateur soit membre de plus d'un groupe. Afin que ces tests ne soient pas sautés, nous allons ajouter un groupe temporaire et y ajouter l'utilisateur `tester` :

```
groupadd -g 102 dummy -U tester
```

Corrigez des droits afin qu'un utilisateur non-`root` puisse compiler et exécuter les tests :

```
chown -R tester .
```

Maintenant, exécutez les tests (en utilisant `/dev/null` comme entrée standard, sinon deux tests peuvent être cassés si vous construisez LFS dans un terminal graphique, dans une session SSH ou via GNU Screen comme l'entrée standard est connectée à un PTY de la distribution hôte, et que le nœud de périphérique de ce PTY n'est pas accessible depuis l'environnement chroot de LFS) :

```
su tester -c "PATH=$PATH make -k RUN_EXPENSIVE_TESTS=yes check" \
   < /dev/null
```

Supprimez le groupe temporaire :

```
groupdel dummy
```

Installez le paquet :

```
make install
```

Déplacez certains programmes aux emplacements spécifiés par le FHS :

```
mv -v /usr/bin/chroot /usr/sbin
mv -v /usr/share/man/man1/chroot.1 /usr/share/man/man8/chroot.8
sed -i 's/"1"/"8"/' /usr/share/man/man8/chroot.8
```

8.59.2. Contenu de Coreutils

Programmes installés: [, b2sum, base32, base64, basename, basenc, cat, chcon, chgrp, chmod, chown, chroot, cksum, comm, cp, csplit, cut, date, dd, df, dir, dircolors, dirname, du, echo, env, expand, expr, factor, false, fmt, fold, groups, head, hostid, id, install, join, link, ln, logname, ls, md5sum, mkdir, mkfifo, mknod, mktemp, mv, nice, nl, nohup, nproc, numfmt, od, paste, pathchk, pinky, pr, printenv, printf, ptx, pwd, readlink, realpath, rm, rmdir, runcon, seq, sha1sum, sha224sum, sha256sum, sha384sum, sha512sum, shred, shuf, sleep, sort, split, stat, stdbuf, stty, sum, sync, tac, tail, tee, test, timeout, touch, tr, true, truncate, tsort, tty, uname, unexpand, uniq, unlink, users, vdir, wc, who, whoami et yes
Bibliothèque installée: libstdbuf.so (dans /usr/libexec/coreutils)
Répertoire installé: /usr/libexec/coreutils

Descriptions courtes

[La commande /usr/bin/[est une vraie commande. Il s'agit d'une équivalente de la commande **test**

base32 Encode et décode des données selon la spécification de la base32 (RFC 4648)

base64 Encode et décode des données selon la spécification de la base64 (RFC 4648)

b2sum Affiche ou vérifie des sommes de contrôle 512-bit BLAKE2

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

basenc Encode ou décode des données avec divers algorithmes

cat Concatène des fichiers sur la sortie standard

chcon Modifie le contexte de sécurité des fichiers et des dossiers

chgrp Change le groupe propriétaire de certains fichiers et répertoires

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

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

chroot Lance une commande avec le répertoire spécifié en tant que répertoire racine (`/`)

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

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

cp Copie des fichiers

csplit Divise un fichier donné en plusieurs fichiers. Les nouveaux fichiers sont séparés par des motifs donnés ou des numéros de lignes et le nombre total d'octets de chaque nouveau fichier est affiché

cut Affiche des parties de lignes et sélectionne ces parties suivant des champs ou positions donnés

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

dd Copie un fichier en utilisant la taille et le nombre de blocs donnés tout en réalisant des conversions optionnelles

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

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

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

dirname Extrait les parties d'un répertoire des noms donnés

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

echo Affiche les chaînes données

env Lance une commande dans un environnement modifié

expand Convertit les tabulations en espaces

expr Évalue des expressions

factor Affiche les facteurs premiers des entiers spécifiés

false Ne fait rien, comme prévu, et renvoie toujours un code d'erreur indiquant l'échec

fmt Reformate les paragraphes dans les fichiers donnés

fold Emballe les lignes des fichiers donnés

groups Affiche les groupes auxquels appartient un utilisateur

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

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

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

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

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

link Crée un lien matériel, avec le nom donné, vers un fichier

ln Crée des liens symboliques ou matériels entre des fichiers

logname Indique l'identifiant de l'utilisateur actuel

ls Liste le contenu de chaque répertoire donné

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

mkdir Crée des répertoires avec les noms donnés

mkfifo Crée des fichiers FIFO (First-In, First-Out), des « tubes nommés » dans le jargon Unix, avec les noms donnés

mknod Crée des nœuds de périphériques avec les noms donnés. Un nœud de périphérique est de type caractère, bloc, ou FIFO

mktemp Crée des fichiers temporaires de manière sécurisée, il est utilisé dans des scripts

mv Déplace ou renomme des fichiers ou répertoires

nice Lance un programme avec une priorité modifiée

nl Numérote les lignes de fichiers donnés

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

nproc Affiche le nombre d'unités d'action disponibles pour un processus

numfmt Convertit des numéros en chaînes lisibles par un humain ou vice-versa

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

paste Joint les fichiers donnés en plaçant les lignes correspondantes l'une à côté de l'autre, en les séparant par des caractères de tabulation

pathchk Vérifie que les noms de fichier sont valides ou portables

pinky Un client finger léger qui affiche certaines informations sur les utilisateurs indiqués

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

printenv Affiche l'environnement

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

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

pwd Indique le nom du répertoire courant

readlink Indique la valeur du lien symbolique

realpath Affiche le chemin résolu

rm Supprime des fichiers ou des répertoires

rmdir Supprime les répertoires vides

runcon Lance une commande avec le contexte de sécurité spécifié

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

sha1sum Affiche ou vérifie des sommes de contrôle 160-bit Secure Hash Algorithm (SHA1)

sha224sum Affiche ou vérifie des sommes de contrôle 224-bit Secure Hash Algorithm (SHA1)

sha256sum Affiche ou vérifie des sommes de contrôle 256-bit Secure Hash Algorithm (SHA1)

sha384sum Affiche ou vérifie des sommes de contrôle 384-bit Secure Hash Algorithm (SHA1)

sha512sum Affiche ou vérifie des sommes de contrôle 512-bit Secure Hash Algorithm (SHA1)

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

shuf Mélange des lignes de texte

sleep Fait une pause d'un certain temps

sort Trie les lignes des fichiers donnés

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

stat Affiche le statut du fichier ou du système de fichiers

stdbuf Lance des commandes avec des opérations de mise en tampon différentes pour ses flux standards

stty Initialise ou affiche les paramètres de la ligne de terminal

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

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

tac Concatène les fichiers donnés à l'envers

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

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

test Compare des valeurs et vérifie les types de fichiers

timeout Lance une commande avec une limite de temps

touch Modifie l'horodatage d'un fichier, initialise les dates et les heures d'accès et de modification des fichiers indiqués à l'heure actuelle. Les fichiers inexistants sont créés avec une longueur nulle

tr Convertit, compresse et supprime les caractères lus depuis l'entrée standard

true Ne fait rien, comme prévu, et quitte toujours avec un code de sortie indiquant une réussite

truncate Réduit ou augmente un fichier selon la taille spécifiée

tsort Réalise un tri topologique et écrit une liste totalement ordonnée suivant un fichier donné partiellement ordonné

tty Indique le nom du fichier du terminal connecté à l'entrée standard

uname Affiche des informations système

unexpand Convertit les espaces en tabulations

uniq Ne conserve qu'une seule ligne parmi plusieurs lignes successives identiques

unlink Supprime le fichier donné

users Indique les noms des utilisateurs actuellement connectés

vdir Est identique à **ls -l**

wc Signale le nombre de lignes, mots et octets de chaque fichier indiqué ainsi que le grand total lorsque plus d'un fichier est donné

who Indique qui est connecté

whoami Indique le nom de l'utilisateur associé avec l'identifieur utilisateur effectif

yes Affiche indéfiniment `y`, ou la chaîne précisée, jusqu'à ce que le processus s'arrête

`libstdbuf` Bibliothèque utilisée par **stdbuf**

8.60. Diffutils-3.12

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

Temps de construction approximatif: 0,5 SBU
Espace disque requis: 51 Mo

8.60.1. Installation de Diffutils

Préparez la compilation de Diffutils :

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

Compilez le paquet :

```
make
```

Pour tester les résultats, exécutez :

```
make check
```

Installez le paquet :

```
make install
```

8.60.2. Contenu de Diffutils

Programmes installés: cmp, diff, diff3 et sdiff

Descriptions courtes

cmp Compare deux fichiers et indique les différences octet par octet
diff Compare deux fichiers ou répertoires et rapporte les lignes où les fichiers diffèrent
diff3 Compare trois fichiers ligne par ligne
sdiff Assemble deux fichiers et affiche le résultat de façon interactive

8.61. Gawk-5.3.2

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

Temps de construction approximatif: 0,2 SBU
Espace disque requis: 45 Mo

8.61.1. Installation de Gawk

Tout d'abord, assurez-vous que certains fichiers inutiles ne sont pas installés :

```
sed -i 's/extras//' Makefile.in
```

Préparez la compilation de Gawk :

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

Compilez le paquet :

```
make
```

Pour tester les résultats, exécutez :

```
chown -R tester .
su tester -c "PATH=$PATH make check"
```

Installez le paquet :

```
rm -f /usr/bin/gawk-5.3.2
make install
```

Signification de la commande :

rm -f /usr/bin/gawk-5.3.2
Le système de construction va recréer le lien en dur `gawk-5.3.2` s'il existe déjà. Supprimez-le pour garantir que le précédent lien en dur installé sur Section 6.9, « Gawk-5.3.2 » soit actualisé ici.

Le processus d'installation a déjà créé **awk** comme lien symbolique vers **gawk**, créez également la page de manuel avec un lien symbolique :

```
ln -sv gawk.1 /usr/share/man/man1/awk.1
```

Si vous le souhaitez, installez la documentation :

```
install -vDm644 doc/{awkforai.txt,*.{eps,pdf,jpg}} -t /usr/share/doc/gawk-5.3.2
```

8.61.2. Contenu de Gawk

Programmes installés: awk (lien vers gawk), gawk et awk-5.3.2
Bibliothèques installées: filefuncs.so, fnmatch.so, fork.so, inplace.so, intdiv.so, ordchr.so, readdir.so, readfile.so, revoutput.so, revtwoway.so, rwarray.so, et time.so (toutes dans /usr/lib/gawk)
Répertoires installés: /usr/lib/gawk, /usr/libexec/awk, /usr/share/awk, et /usr/share/doc/gawk-5.3.2

Descriptions courtes

awk Un lien vers **gawk**
gawk Un programme de manipulation de fichiers texte. C'est l'implémentation GNU d'**awk**
gawk-5.3.2 Un lien matériel vers **gawk**

8.62. Findutils-4.10.0

Le paquet Findutils contient des programmes de recherche de fichiers. Ces programmes sont fournis pour parcourir tous les fichiers dans une hiérarchie de répertoires et pour créer, maintenir et parcourir 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). Findutils fournit également le programme **xargs**, qui peut être utilisé pour exécuter une commande spécifique sur chaque fichier sélectionné par la recherche.

Temps de construction approximatif: 0,7 SBU
Espace disque requis: 61 Mo

8.62.1. Installation de Findutils

Préparez la compilation de Findutils :

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

Voici la signification des options de configuration :

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

Compilez le paquet :

```
make
```

Pour tester les résultats, exécutez :

```
chown -R tester .
su tester -c "PATH=$PATH make check"
```

Installez le paquet :

```
make install
```

8.62.2. Contenu de Findutils

Programmes installés: find, locate, updatedb et xargs
Répertoire installé: /var/lib/locate

Descriptions courtes

find Cherche dans les hiérarchies de répertoires donnés les fichiers correspondant à un critère spécifié

locate Recherche à travers la base de données des noms de fichiers et renvoie ceux qui contiennent une certaine chaîne ou qui correspondent à un certain modèle

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

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

8.63. Groff-1.23.0

Le paquet Groff contient des programmes pour le traitement et la mise en forme des textes et des images.

Temps de construction approximatif: 0,2 SBU
Espace disque requis: 108 Mo

8.63.1. Installation de Groff

Groff s'attend à ce que la variable d'environnement `PAGE` contienne la taille du papier par défaut. Pour les utilisateurs américains, `PAGE=letter` est adéquate. `PAGE=A4` pourrait aller mieux ailleurs. Même si la taille du papier par défaut est configurée lors de la compilation, elle peut être réécrite plus tard en écrivant « A4 » ou « letter » dans le fichier `/etc/papersize`.

Préparez la compilation de Groff :

```
PAGE=<taille_papier> ./configure --prefix=/usr
```

Construisez le paquet :

```
make
```

Pour tester les résultats, exécutez :

```
make check
```

Installez le paquet :

```
make install
```

8.63.2. Contenu de Groff

Programmes installés: addftinfo, afmtodit, chem, eqn, eqn2graph, gdiffmk, glilypond, gperl, gpinyin, grap2graph, grn, grodvi, groff, groffer, grog, grolbp, grolj4, gropdf, grops, grotty, hpftodit, indxbib, lkbib, lookbib, mmroff, neqn, nroff, pdfmom, pdfroff, pfbtops, pic, pic2graph, post-grohtml, preconv, pre-grohtml, refer, roff2dvi, roff2html, roff2pdf, roff2ps, roff2text, roff2x, soelim, tbl, tfmtodit, et troff
Répertoires installés: /usr/lib/groff and /usr/share/doc/groff-1.23.0, /usr/share/groff

Descriptions courtes

addftinfo Lit un fichier de polices troff et ajoute quelques informations métriques supplémentaires sur la police qui est utilisée par le système **groff**

afmtodit Crée un fichier de police à utiliser avec **groff** et **grops**

chem Préprocesseur Groff pour produire des diagrammes de structure chimique

eqn Compile les descriptions d'équations imbriquées dans les fichiers d'entrée de troff pour obtenir des commandes comprises par **troff**

eqn2graph Convertit une équation EQN troff en une image recadrée

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

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

gperl Préprocesseur de groff permettant d'ajouter du code perl dans les fichiers groff

gpinyin Préprocesseur de groff permettant d'ajouter du Pinyin (mandarin épelé avec l'alphabet latin) dans les fichiers groff.

grap2graph Convertit un fichier grap en une image matricielle recadrée (grap est un ancien langage de programmation Unix dédié à la création de diagrammes)

grn Un préprocesseur **groff** pour les fichiers gremlin

grodvi Un pilote pour **groff** qui produit des fichiers avec le format de sortie dvi TeX

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

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

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

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

grolj4 Est un pilote pour **groff** produisant une sortie au format PCL5 adapté aux imprimantes HP Laserjet 4

gropdf Traduit la sortie de GNU **troff** en PDF

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

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

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

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

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

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

mmroff Un simple pré-processeur pour **groff**

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

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

pdfmom Est une enveloppe autour de groff qui facilite la production de documents PDF depuis des fichiers formattés avec les macros parentes.

pdfroff Crée des documents pdf en utilisant groff

pfbtops Traduit une police Postscript au format `.pfb` vers le format ASCII

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

pic2graph Convertit un diagramme PIC en une image recadrée

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

preconv Convertit l'encodage de fichiers en entrée vers quelque chose que comprend GNU **troff**

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

refer	Copie le contenu d'un fichier sur la sortie standard, sauf pour les lignes entre les symboles *.[* et *.]* interprétées comme des citations, et les lignes entre *.R1* et *.R2* interprétées comme des commandes sur la façon de gérer les citations
roff2dvi	Transforme des fichiers roff en fichiers DVI
roff2html	Transforme les fichiers roff en fichiers HTML
roff2pdf	Transforme les fichiers roff en fichiers PDF
roff2ps	Transforme les fichiers roff en fichiers ps
roff2text	Transforme les fichiers roff en fichiers textes
roff2x	Transforme les fichiers roff vers d'autres formats
soelim	Lit des fichiers et remplace les lignes de la forme *.so fichier* par le contenu du *fichier* mentionné
tbl	Compile les descriptions des tables imbriquées dans les fichiers d'entrées troff en commandes comprises par **troff**
tfmtodit	Crée un fichier de police à utiliser avec **groff -Tdvi**
troff	Est hautement compatible avec la commande Unix **troff**. Habituellement, il devrait être appelé en utilisant la commande **groff** qui lance aussi les pré-processeurs et post-processeurs dans l'ordre approprié et avec les options appropriées

8.64. GRUB-2.12

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

Temps de construction approximatif: 0,3 SBU
Espace disque requis: 166 Mo

8.64.1. Installation de GRUB

Note

Si votre système prend en charge l'UEFI et que vous souhaitez démarrer LFS avec l'UEFI, vous devez installer GRUB avec la prise en charge de l'UEFI (et ses dépendances) en suivant les instructions de *la page BLFS* à la fin de ce chapitre. Vous pouvez passer ce paquet ou installer ce paquet et le paquet GRUB de BLFS pour UEFI sans conflit (la page de BLFS propose des instructions pour les deux cas).

Avertissement

Déconfigurez les variables d'environnement qui pourrait perturber la construction :

```
unset {C,CPP,CXX,LD}FLAGS
```

N'essayez pas de « customiser » ce paquet avec des options de compilation personnalisées. Ce paquet est un chargeur de démarrage. Un optimisation agressive du paquet pourrait casser les opérations de bas-niveau dans le code source.

Ajoutez un fichier qui manque dans l'archive publiée :

```
echo depends bli part_gpt > grub-core/extra_deps.lst
```

Préparez la compilation de GRUB :

```
./configure --prefix=/usr     \
            --sysconfdir=/etc \
            --disable-efiemu  \
            --disable-werror
```

Voici la signification des nouvelles options de configure :

`--disable-werror`
Cette option permet de terminer la compilation avec les avertissements ajoutés dans des versions de Flex plus récentes.

`--disable-efiemu`
Cette option désactive les fonctionnalités et des programmes de tests non nécessaires pour LFS pour minimiser la construction.

Compilez le paquet :

```
make
```

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

Installez le paquet et déplacez le fichier de prise en charge de l'autocomplétion Bash vers l'emplacement recommandé par les mainteneurs de l'autocomplétion de Bash :

```
make install
mv -v /etc/bash_completion.d/grub /usr/share/bash-completion/completions
```

Le chapitre Section 10.4, « Utiliser GRUB pour paramétrer le processus de démarrage » explique comment permettre au système LFS de démarrer avec GRUB.

8.64.2. Contenu de GRUB

Programmes installés: grub-bios-setup, grub-editenv, grub-file, grub-fstest, grub-glue-efi, grub-install, grub-kbdcomp, grub-macbless, grub-menulst2cfg, grub-mkconfig, grub-mkimage, grub-mklayout, grub-mknetdir, grub-mkpasswd-pbkdf2, grub-mkrelpath, grub-mkrescue, grub-mkstandalone, grub-ofpathname, grub-probe, grub-reboot, grub-render-label, grub-script-check, grub-set-default, grub-sparc64-setup et grub-syslinux2cfg

Répertoires installés: /usr/lib/grub, /etc/grub.d, /usr/share/grub et boot/grub (après avoir lancé grub-install pour la première fois)

Descriptions courtes

grub-bios-setup	Programme d'aide pour **grub-install**
grub-editenv	Est un outil qui permet d'éditer le bloc d'environnement
grub-file	Vérifie que le type du fichier sélectionné est celui attendu
grub-fstest	Est un outil de débogage du pilote d'un système de fichiers
grub-glue-efi	Assemble des binaires 32-bits et 64-bits en un seul fichier (pour les ordinateurs Apple)
grub-install	Installe GRUB sur votre disque
grub-kbdcomp	Est un script qui convertit un plan xkb dans un plan reconnu par GRUB
grub-macbless	Est l'équivalent Mac de la commande bless pour les systèmes de fichiers HFS ou HFS+ (la commande **bless** est exclusive au ordinateurs Apple : elle permet au système de démarrer)
grub-menulst2cfg	Convertit un `menu.lst` du GRUB de base en fichier `grub.cfg` utilisable avec GRUB 2
grub-mkconfig	Génère un fichier `grub.cfg`
grub-mkimage	Crée une image GRUB démarrable
grub-mklayout	Génère un fichier de plan de clavier pour GRUB
grub-mknetdir	Prépare un répertoire GRUB d'amorçage par le réseau
grub-mkpasswd-pbkdf2	Génère un mot de passe PBKDF2 chiffré pour une utilisation dans le menu de démarrage
grub-mkrelpath	Rend relatif le nom de chemin vers la racine d'un système
grub-mkrescue	Crée une image GRUB démarrable adaptée à une disquette, à un CDROM/DVD ou à un clé USB
grub-mkstandalone	Génère une image autonome
grub-ofpathname	Est un programme d'aide qui affiche le chemin d'un périphérique GRUB
grub-probe	Teste les informations de périphérique pour un chemin ou un périphérique donné
grub-reboot	Règle l'entrée d'amorçage par défaut pour GRUB uniquement pour le prochain démarrage
grub-render-label	Produit des .disk_label Apple pour les Macs Apple
grub-script-check	Cherche les erreurs de syntaxe dans le script de configuration de GRUB
grub-set-default	Règle l'entrée d'amorçage par défaut pour GRUB

grub-sparc64-setup	Est un programme d'aide pour grub-setup
grub-syslinux2cfg	Transforme un fichier de configuration syslinux vers le format de grub.cfg

8.65. Gzip-1.14

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

Temps de construction approximatif: 0,1 SBU
Espace disque requis: 21 Mo

8.65.1. Installation de Gzip

Préparez la compilation de Gzip :

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

Compilez le paquet :

```
make
```

Pour tester les résultats, exécutez :

```
make check
```

Installez le paquet :

```
make install
```

8.65.2. Contenu de Gzip

Programmes installés: gunzip, gzexe, gzip, uncompress (lien matériel vers gunzip), zcat, zcmp, zdiff, zegrep, zfgrep, zforce, zgrep, zless, zmore et znew

Descriptions courtes

gunzip Décompresse les fichiers gzip

gzexe Crée des fichiers exécutables auto-extractibles

gzip Compresse les fichiers donnés en utilisant le codage Lempel-Ziv (LZ77)

uncompress Décompresse les fichiers compressés

zcat Décompresse les fichiers gzip sur la sortie standard

zcmp Lance **cmp** sur des fichiers compressés avec gzip

zdiff Lance **diff** sur des fichiers compressés avec gzip

zegrep Lance **egrep** sur des fichiers compressés avec gzip

zfgrep Lance **fgrep** sur des fichiers compressés avec gzip

zforce Force une extension `.gz` sur tous les fichiers donnés qui sont au format gzip, pour que **gzip** ne les compresse pas de nouveau ; ceci est utile quand les noms de fichiers sont tronqués lors d'un transfert de fichiers

zgrep Lance **grep** sur des fichiers compressés avec gzip

zless Lance **less** sur des fichiers compressés avec gzip

zmore Lance **more** sur des fichiers compressés avec gzip

znew Recompresse les fichiers formatés avec **compress** au format **gzip**— de `.z` vers `.gz`

8.66. IPRoute2-6.16.0

Le paquet IPRoute2 contient des programmes pour le réseau, basique ou avancé, basé sur IPV4.

Temps de construction approximatif: 0,1 SBU
Espace disque requis: 17 Mo

8.66.1. Installation de IPRoute2

Le programme **arpd** inclus dans ce paquet ne sera pas chargé car il dépend de Berkeley DB et ce dernier n'est pas installé dans LFS. Un répertoire et une page de manuel pour **arpd** seront tout de même installés. Empêchez-le en exécutant la commande suivante.

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

Compilez le paquet :

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

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

Installez le paquet :

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

Si vous le souhaitez, installez la documentation :

```
install -vDm644 COPYING README* -t /usr/share/doc/iproute2-6.16.0
```

8.66.2. Contenu de IPRoute2

Programmes installés: bridge, ctstat (lien vers lnstat), genl, ifstat, ip, lnstat, nstat, routel, rtacct, rtmon, rtpr, rtstat (lien vers lnstat), ss et tc
Répertoires installés: /etc/iproute2, /usr/lib/tc et /usr/share/doc/iproute2-6.16.0

Descriptions courtes

bridge Configure des ponts réseaux

ctstat Outil donnant le statut de la connexion

genl Interface utilitaire du générique Netlink

ifstat Affiche les statistiques d'une interface, incluant le nombre de paquets transmis et reçus par l'interface

ip L'exécutable principal. Il a plusieurs fonctions différentes dont :
ip link ***<périphérique>*** autorise les utilisateurs à regarder l'état des périphériques et à faire des changements
ip addr autorise les utilisateurs à regarder les adresses et leurs propriétés, à ajouter de nouvelles adresses et à supprimer les anciennes
ip neighbor autorise les utilisateurs à regarder dans les liens des voisins et dans leurs propriétés, à ajouter de nouvelles entrées et à supprimer les anciennes
ip rule autorise les utilisateurs à regarder les politiques de routage et à les modifier
ip route autorise les utilisateurs à regarder la table de routage et à modifier les règles de routage
ip tunnel autorise les utilisateurs à regarder les tunnels IP et leurs propriétés, et à les modifier
ip maddr autorise les utilisateurs à regarder les adresses multicast et leurs propriétés, et à les changer
ip mroute autorise les utilisateurs à configurer, modifier ou supprimer le routage multicast

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

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

nstat Affiche les statistiques du réseau

routel Un composant de **ip route** qui affiche les tables de routage

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

rtmon Outil de surveillance de routes

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

rtstat Outil de statut de routes

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

tc 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 mise en file d'attente
tc class autorise les utilisateurs à configurer les classes suivant la planification de la mise en file d'attente
tc filter autorise les utilisateurs à configurer les filtres de paquets QOS/COS
tc monitor peut être utilisé pour voir les modifications faites au contrôle de trafic dans le noyau.

8.67. Kbd-2.8.0

Le paquet Kbd contient les fichiers de tables de caractères, les polices de la console et des outils pour le clavier.

Temps de construction approximatif: 0,1 SBU
Espace disque requis: 43 Mo

8.67.1. Installation de Kbd

Le comportement des touches Retour Arrière et Supprimer n'est pas cohérent dans toutes les tables de correspondance du clavier du paquet Kbd. Le correctif suivant répare ce problème pour les tables de correspondance i386 du clavier :

```
patch -Np1 -i ../kbd-2.8.0-backspace-1.patch
```

Après la correction, la touche Retour Arrière génère le caractère de code 127, et la touche Supprimer génère une séquence d'échappement bien connue.

Supprimez le programme **resizecons** redondant (il exige feu svgalib pour fournir les fichiers du mode graphique - pour une utilisation normale, **setfont** redimensionne correctement la console) ainsi que sa page de man.

```
sed -i '/RESIZECONS_PROGS=/s/yes/no/' configure
sed -i 's/resizecons.8 //' docs/man/man8/Makefile.in
```

Préparez la compilation de Kbd :

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

Voici la signification de l'option de configuration :

`--disable-vlock`
Cette option empêche la construction de l'utilitaire vlock car il requiert la bibliothèque PAM qui n'est pas disponible dans l'environnement chroot.

Compilez le paquet :

```
make
```

Les tests de ce paquet échoueront tous dans l'environnement chroot car ils nécessitent valgrind. De plus, sur un système complet avec valgrind, plusieurs tests échouent toujours dans un environnement graphique. Les tests passent dans un environnement non graphique.

Installez le paquet :

```
make install
```

Note

Pour certaines langues (comme le biélorusse), le paquet Kbd ne fournit pas une table de correspondance utile, puisque le contenu de la table « by » suppose l'encodage ISO-8859-5, et la table CP1251 est normalement utilisée. Les utilisateurs de telles langues doivent télécharger les tables de correspondance qui conviennent séparément.

Si vous le souhaitez, installez la documentation :

```
cp -R -v docs/doc -T /usr/share/doc/kbd-2.8.0
```

8.67.2. Contenu de Kbd

Programmes installés: chvt, deallocvt, dumpkeys, fgconsole, getkeycodes, kbdinfo, kbd_mode, kbdrate, loadkeys, loadunimap, mapscrn, openvt, psfaddtable (lien vers psfxtable), psfgettable (lien vers psfxtable), psfstriptable (lien vers psfxtable), psfxtable, setfont, setkeycodes, setleds, setmetamode, setvtrgb, showconsolefont, showkey, unicode_start et unicode_stop

Répertoires installés: /usr/share/consolefonts, /usr/share/consoletrans, /usr/share/keymaps, /usr/share/doc/kbd-2.8.0 et /usr/share/unimaps

Descriptions courtes

chvt	Change le terminal virtuel en avant plan
deallocvt	Désalloue les terminaux virtuels inutilisés
dumpkeys	Affiche la table de traduction du clavier
fgconsole	Affiche le numéro du terminal virtuel actif
getkeycodes	Affiche la table de correspondance des « scancode » avec les « keycode »
kbdinfo	Obtient des informations sur l'état d'une console
kbd_mode	Affiche ou initialise le mode du clavier
kbdrate	Initialise les taux de répétition et de délai du clavier
loadkeys	Charge les tables de traduction du clavier
loadunimap	Charge la table de correspondance du noyau unicode-police
mapscrn	Un programme obsolète utilisé pour charger une table de correspondance des caractères de sortie définie par l'utilisateur dans le pilote de la console. Ceci est maintenant fait par **setfont**
openvt	Lance un programme sur un nouveau terminal virtuel (VT)
psfaddtable	Ajoute une table de caractères Unicode à la police d'une console
psfgettable	Extrait la table de caractères Unicode embarquée dans la police de la console
psfstriptable	Supprime la table de caractères Unicode embarquée dans la police de la console
psfxtable	Gère les tables de caractères Unicode pour les polices de la console
setfont	Modifie les polices EGA/VGA (*Enhanced Graphic Adapter-Video Graphics Array*) sur la console
setkeycodes	Charge les entrées de la table de correspondance entre scancode et keycode, utile si vous avez des touches inhabituelles sur votre clavier
setleds	Initialise les drapeaux et LED du clavier
setmetamode	Définit la gestion de la touche méta du clavier
setvtrgb	Définit la table de couleur console dans tous les terminaux virtuels
showconsolefont	Affiche la police de l'écran pour la console EGA/VGA
showkey	Affiche les scancodes, keycodes et codes ASCII des touches appuyées sur le clavier
unicode_start	Met le clavier et la console en mode UNICODE. N'utilisez pas ce programme sauf si votre fichier de correspondance est encodé en ISO-8859-1. Pour les autres encodages, cet utilitaire donne de mauvais résultats.
unicode_stop	Ramène le clavier et la console dans le mode avant UNICODE

8.68. Libpipeline-1.5.8

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

Temps de construction approximatif: 0,1 SBU
Espace disque requis: 10 Mo

8.68.1. Installation de Libpipeline

Préparez la compilation de Libpipeline :

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

Compilez le paquet :

```
make
```

Les tests nécessitent la bibliothèque Check que nous avons supprimée le LFS.

Installez le paquet :

```
make install
```

8.68.2. Contenu de Libpipeline

Bibliothèque installée: libpipeline.so

Descriptions courtes

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

8.69. Make-4.4.1

Le paquet Make contient un programme pour contrôler la génération d'exécutables et d'autres fichiers non-sources d'un paquet à partir des fichiers sources.

Temps de construction approximatif: 0,7 SBU
Espace disque requis: 13 Mo

8.69.1. Installation de Make

Préparez la compilation de Make :

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

Compilez le paquet :

```
make
```

Pour tester les résultats, exécutez :

```
chown -R tester .
su tester -c "PATH=$PATH make check"
```

Installez le paquet :

```
make install
```

8.69.2. Contenu de Make

Programme installé: make

Descriptions courtes

make Détermine automatiquement quelles pièces d'un paquet doivent être (re)compilées. Puis, il lance les commandes adéquates

8.70. Patch-2.8

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

Temps de construction approximatif: 0,2 SBU
Espace disque requis: 13 Mo

8.70.1. Installation de Patch

Préparez la compilation de Patch :

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

Compilez le paquet :

```
make
```

Pour tester les résultats, exécutez :

```
make check
```

Installez le paquet :

```
make install
```

8.70.2. 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.

8.71. Tar-1.35

Le paquet Tar fournit la possibilité de créer des archives tar et effectuer diverses manipulations d'archives. Tar peut être utilisé sur des archives précédemment créées pour extraire des fichiers, ajouter des fichiers supplémentaires, mettre à jour ou lister les fichiers qui étaient déjà stockés.

Temps de construction approximatif: 0,6 SBU
Espace disque requis: 43 Mo

8.71.1. Installation de Tar

Préparez la compilation de Tar :

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

Voici la signification de l'option de configuration :

`FORCE_UNSAFE_CONFIGURE=1`
Ceci oblige le test de `mknod` à se lancer en tant que `root`. On considère généralement que lancer ce test en tant qu'utilisateur `root` est dangereux, mais comme on ne l'exécute que sur un système qui n'a été construit que partiellement, ce remplacement est acceptable.

Compilez le paquet :

```
make
```

Pour tester les résultats, exécutez :

```
make check
```

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

Installez le paquet :

```
make install
make -C doc install-html docdir=/usr/share/doc/tar-1.35
```

8.71.2. Contenu de Tar

Programmes installés: tar
Répertoire installé: /usr/share/doc/tar-1.35

Descriptions courtes

tar Crée, extrait des fichiers à partir d'archives et liste le contenu d'archives, connues sous le nom d'archives tar

8.72. Texinfo-7.2

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

Temps de construction approximatif: 0,4 SBU
Espace disque requis: 160 Mo

8.72.1. Installation de Texinfo

Corrigez une manière de coder qui fait afficher un avertissement à Perl-5.42 et supérieur :

```
sed 's/! $output_file eq/$output_file ne/' -i tp/Texinfo/Convert/*.pm
```

Préparez la compilation de Texinfo :

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

Compilez le paquet :

```
make
```

Pour tester les résultats, exécutez :

```
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
```

Signification des paramètres de make :

TEXMF=/usr/share/texmf
La variable TEXMF du Makefile contient l'emplacement de la racine de votre répertoire TeX si, par exemple, un paquet TeX est installé plus tard.

Le système de documentation Info utilise un fichier texte pour contenir sa liste des entrées de menu. Le fichier est situé dans /usr/share/info/dir. Malheureusement, à cause de problèmes occasionnels dans les Makefile de différents paquets, il peut être non synchronisé avec les pages info. Si le fichier /usr/share/info/dir a besoin d'être recréé, les commandes suivantes accompliront cette tâche :

```
pushd /usr/share/info
  rm -v dir
  for f in *
    do install-info $f dir 2>/dev/null
  done
popd
```

8.72.2. Contenu de Texinfo

Programmes installés: info, install-info, makeinfo (lien vers texi2any), pdftexi2dvi, pod2texi, texi2any, texi2dvi, texi2pdf et texindex
Bibliothèque installée: MiscXS.so, Parsetexi.so et XSParagraph.so (le tout dans /usr/lib/texinfo)
Répertoires installés: /usr/share/texinfo et /usr/lib/texinfo

Descriptions courtes

info Utilisé pour lire des pages info similaires aux pages man mais qui vont souvent plus loin que la simple explication des arguments disponibles. Par exemple, comparez **man bison** et **info bison**.

install-info	Utilisé pour installer les pages info ; il met à jour les entrées dans le fichier index d'**info**
makeinfo	Traduit les sources Texinfo données dans différents autres langages : pages info, texte ou HTML
pdftexi2dvi	Utilisé pour formater le document Texinfo indiqué en un fichier PDF (*Portable Document Format*)
pod2texi	Convertit des documents Pod vers le format Texinfo
texi2any	Traduit des documentations source Texinfo vers différents autres formats
texi2dvi	Utilisé pour formater le document Texinfo indiqué en un fichier indépendant des périphériques, pouvant être imprimé
texi2pdf	Utilisé pour formater le document Texinfo indiqué en un fichier PDF (*Portable Document Format*)
texindex	Utilisé pour trier les fichiers d'index de Texinfo

8.73. Vim-9.1.1629

Le paquet Vim contient un puissant éditeur de texte.

Temps de construction approximatif: 3,7 SBU
Espace disque requis: 259 Mo

Alternatives à Vim

Si vous préférez un autre éditeur—comme Emacs, Joe, ou Nano—merci de vous référer à *https://fr.linuxfromscratch.org/blfs/../view/blfs-12.4-fr/postlfs/editors.html* pour des instructions d'installation.

8.73.1. Installation de Vim

Tout d'abord, modifiez l'emplacement par défaut du fichier de configuration `vimrc` en `/etc` :

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

Préparez la compilation de Vim :

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

Compilez le paquet :

```
make
```

Pour préparer les tests, assurez-vous que l'utilisateur `tester` puisse écrire dans l'arborescence des sources et excluez un fichier qui contient des tests qui nécessitent **curl** ou **wget** :

```
chown -R tester .
sed '/test_plugin_glvs/d' -i src/testdir/Make_all.mak
```

Maintenant lancez les tests en tant qu'utilisateur `tester` :

```
su tester -c "TERM=xterm-256color LANG=en_US.UTF-8 make -j1 test" \
   &> vim-test.log
```

La suite de tests affiche à l'écran beaucoup de caractères binaires. Ils peuvent causer des soucis avec les paramètres de votre terminal actuel (surtout quand on remplace la variable `TERM` pour satisfaire des hypothèses de la suite de tests). Le problème peut se résoudre en redirigeant la sortie vers un journal de traces comme montré ci-dessus. Un test réussi affichera les mots `ALL DONE` dans le journal à la fin du processus.

Installez le paquet :

```
make install
```

Beaucoup d'utilisateurs sont habitués à utiliser **vi** au lieu de **vim**. Pour permettre l'exécution de **vim** quand les utilisateurs saisissent habituellement **vi**, créez un lien symbolique vers les binaires et vers les pages de man dans les langues fournies :

```
ln -sv vim /usr/bin/vi
for L in  /usr/share/man/{,*/}man1/vim.1; do
    ln -sv vim.1 $(dirname $L)/vi.1
done
```

Par défaut, la documentation de Vim s'installe dans `/usr/share/vim`. Le lien symbolique suivant permet d'accéder à la documentation via `/usr/share/doc/vim-9.1.1629`, en cohérence avec l'emplacement de la documentation d'autres paquets :

```
ln -sv ../vim/vim91/doc /usr/share/doc/vim-9.1.1629
```

Si vous allez installer le système de fenêtrage X sur votre système LFS, il pourrait être nécessaire de recompiler Vim après avoir installé X. Vim fournit une version graphique de l'éditeur qui requiert X et quelques autres bibliothèques pour s'installer. Pour plus d'informations sur ce processus, référez-vous à la documentation de Vim et à la page d'installation de Vim dans le livre BLFS sur *https://fr.linuxfromscratch.org/blfs/../view/blfs-12.4-fr/postlfs/vim.html*.

8.73.2. Configuration de Vim

Par défaut, **vim** est lancé en mode compatible vi. Ceci pourrait être nouveau pour les personnes qui ont utilisé d'autres éditeurs dans le passé. Le paramètre « nocompatible » est inclus ci-dessous pour surligner le fait qu'un nouveau comportement est en cours d'utilisation. Il rappelle aussi à ceux qui voudraient le changer en mode « compatible » qu'il devrait être le premier paramètre dans le fichier de configuration. Ceci est nécessaire car il modifie d'autres paramètres et la surcharge doit survenir après ce paramètre. Créez un fichier de configuration **vim** par défaut en lançant ce qui suit :

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

" Ensure defaults are set before customizing settings, not after
source $VIMRUNTIME/defaults.vim
let skip_defaults_vim=1

set nocompatible
set backspace=2
set mouse=
syntax on
if (&term == "xterm") || (&term == "putty")
  set background=dark
endif

" Fin de /etc/vimrc
EOF
```

L'option *`set nocompatible`* change le comportement de **vim** d'une façon plus utile (par défaut) que le comportement compatible vi. Supprimez « no » pour conserver l'ancien comportement de **vi**. Le paramètre *`set backspace=2`* permet le retour en arrière après des sauts de ligne, l'indentation automatique et le début de l'insertion. L'instruction *`syntax on`* active la coloration syntaxique. Le paramètre *`set mouse=r`* permet de coller du texte avec la souris correctement dans un environnement chroot ou au travers d'une connexion à distance. Enfin, l'instruction *if* avec *`set background=dark`* corrige l'estimation de **vim** concernant la couleur du fond de certains émulateurs de terminaux. Ceci permet d'utiliser de meilleures gammes de couleurs pour la coloration syntaxique, notamment avec les fonds noirs de ces programmes.

La documentation pour les autres options disponibles peut être obtenue en exécutant la commande suivante :

```
vim -c ':options'
```

Note

Par défaut, Vim installe uniquement les fichiers de dictionnaires pour l'anglais. Pour installer des fichiers de dictionnaires pour votre langue, copiez le fichier `.spl` et éventuellement, le `.sug` pour votre langue et votre encodage de `runtime/spell` vers `/usr/share/vim/vim91/spell/`.

Pour utiliser ces fichiers dictionnaires, il faut une configuration dans `/etc/vimrc`, comme :

```
set spelllang=en,ru
set spell
```

Pour plus d'information, voir `runtime/spell/README.txt`.

8.73.3. Contenu de Vim

Programmes installés: ex (lien vers vim), rview (lien vers vim), rvim (lien vers vim), vi (lien vers vim), view (lien vers vim), vim, vimdiff (lien vers vim), vimtutor, et xxd
Répertoire installé: /usr/share/vim

Descriptions courtes

ex Démarre **vim** en mode ex

rview Une version restreinte de **view** : aucune commande shell ne peut être lancée et **view** ne peut pas être suspendu

rvim Une version restreinte de **vim** : aucune commande shell ne peut être lancée et **vim** ne peut pas être suspendu

vi Lien vers **vim**

view Démarre **vim** en mode lecture seule

vim L'éditeur

vimdiff Édite deux ou trois versions d'un fichier avec **vim** et montre les différences

vimtutor Vous apprend les touches et les commandes basiques de **vim**

xxd Fait un affichage hexa du fichier donné. Il peut aussi faire l'inverse pour une correspondance binaire

8.74. MarkupSafe-3.0.2

MarkupSafe est un module Python qui implémente les chaînes sures pour le balisage XML/HTML/XHTML.

Temps de construction approximatif: moins de 0,1 SBU
Espace disque requis: 500 Ko

8.74.1. Installation de MarkupSafe

Compilez MarkupSafe avec la commande suivante :

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

Ce paquet n'a pas de suite de tests.

Installez le paquet :

```
pip3 install --no-index --find-links dist Markupsafe
```

8.74.2. Contenu de MarkupSafe

Répertoire installé: /usr/lib/python3.13/site-packages/MarkupSafe-3.0.2.dist-info

8.75. Jinja2-3.1.6

Jinja2 est un module Python qui implémente un langage de modèles simple qui ressemble à python.

Temps de construction approximatif: moins de 0,1 SBU
Espace disque requis: 2,6 Mo

8.75.1. Installation de Jinja2

Construisez le paquet :

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

Installez le paquet :

```
pip3 install --no-index --find-links dist Jinja2
```

8.75.2. Contenu de Jinja2

Répertoire installé: /usr/lib/python3.13/site-packages/Jinja2-3.1.6.dist-info

8.76. Udev de Systemd-257.8

Le paquet Udev contient des programmes pour la création dynamique de nœuds de périphérique.

Temps de construction approximatif: 0,1 SBU
Espace disque requis: 162 Mo

8.76.1. Installation d'Udev

Udev fait partie du paquet systemd-257.8. Utilisez le fichier systemd-257.8.tar.xz comme archive des sources.

Supprimez deux groupes inutiles, `render` et `sgx` des règles udev par défaut :

```
sed -e 's/GROUP="render"/GROUP="video"/' \
    -e 's/GROUP="sgx", //'               \
    -i rules.d/50-udev-default.rules.in
```

Supprimez une règle udev requise pour une installation Systemd complète :

```
sed -i '/systemd-sysctl/s/^/#/' rules.d/99-systemd.rules.in
```

Ajustez les chemins codés en dur aux fichiers de configuration réseau pour l’installation udev indépendante :

```
sed -e '/NETWORK_DIRS/s/systemd/udev/' \
    -i src/libsystemd/sd-network/network-util.h
```

Préparez la compilation d'Udev :

```
mkdir -p build
cd       build

meson setup ..                  \
      --prefix=/usr             \
      --buildtype=release       \
      -D mode=release           \
      -D dev-kvm-mode=0660      \
      -D link-udev-shared=false \
      -D logind=false           \
      -D vconsole=false
```

Voici la signification des options meson :

`--buildtype=release`
Ce paramètre remplace le type de construction par défaut (« debug »), qui produit des binaires non optimisés.

`-D mode=release`
Désactive certaines fonctionnalités considérées comme expérimentales par les développeurs en amont.

`-D dev-kvm-mode=0660`
La règle udev par défaut permet à tous les utilisateurs d'accéder à `/dev/kvm`. Les auteurs pensent que cela est dangereux. Cette option change cela.

`-D link-udev-shared=false`
Cette option évite qu'udev ne se lie à la bibliothèque interne partagée de systemd, `libsystemd-shared`. Cette bibliothèque est conçue pour être partagée par plusieurs composants Systemd ce qui est excessif poru une installation d'udev seulement.

`-D logind=false -D vconsole=false`
Ces options évitent la génération de plusieurs fichiers de règles udev qui appartiennent aux autres composants Systemd qui ne seront pas installés.

Récupérez la liste des outils d'aide empaquetés avec udev et sauvegardez-la dans une variable d'environnement (l'exporter n'est pas absolument nécessaire, mais cela facilite la construction en tant qu'utilisateur ordinaire et l'utilisation d'un gestionnaire de paquet) :

```
export udev_helpers=$(grep "'name' :" ../src/udev/meson.build | \
                      awk '{print $3}' | tr -d ",'" | grep -v 'udevadm')
```

Ne construisez que les composants nécessaires pour udev :

```
ninja udevadm systemd-hwdb                                          \
      $(ninja -n | grep -Eo '(src/(lib)?udev|rules.d|hwdb.d)/[^ ]*') \
      $(realpath libudev.so --relative-to .)                        \
      $udev_helpers
```

Installez le paquet :

```
install -vm755 -d {/usr/lib,/etc}/udev/{hwdb.d,rules.d,network}
install -vm755 -d /usr/{lib,share}/pkgconfig
install -vm755 udevadm                             /usr/bin/
install -vm755 systemd-hwdb                        /usr/bin/udev-hwdb
ln      -svfn  ../bin/udevadm                      /usr/sbin/udevd
cp      -av    libudev.so{,*[0-9]}                 /usr/lib/
install -vm644 ../src/libudev/libudev.h            /usr/include/
install -vm644 src/libudev/*.pc                    /usr/lib/pkgconfig/
install -vm644 src/udev/*.pc                       /usr/share/pkgconfig/
install -vm644 ../src/udev/udev.conf               /etc/udev/
install -vm644 rules.d/* ../rules.d/README         /usr/lib/udev/rules.d/
install -vm644 $(find ../rules.d/*.rules \
                      -not -name '*power-switch*') /usr/lib/udev/rules.d/
install -vm644 hwdb.d/*  ../hwdb.d/{*.hwdb,README} /usr/lib/udev/hwdb.d/
install -vm755 $udev_helpers                       /usr/lib/udev
install -vm644 ../network/99-default.link          /usr/lib/udev/network
```

Installez certaines règles personnalisées et autres fichiers auxiliaires utiles dans un environnement LFS :

```
tar -xvf ../../udev-lfs-20230818.tar.xz
make -f udev-lfs-20230818/Makefile.lfs install
```

Installez les pages de manuel :

```
tar -xf ../../systemd-man-pages-257.8.tar.xz                            \
    --no-same-owner --strip-components=1                                \
    -C /usr/share/man --wildcards '*/udev*' '*/libudev*'                \
                                  '*/systemd.link.5'                    \
                                  '*/systemd-'{hwdb,udevd.service}.8

sed 's|systemd/network|udev/network|'                                   \
    /usr/share/man/man5/systemd.link.5                                  \
  > /usr/share/man/man5/udev.link.5

sed 's/systemd\(\\?-\)/udev\1/' /usr/share/man/man8/systemd-hwdb.8      \
                              > /usr/share/man/man8/udev-hwdb.8

sed 's|lib.*udevd|sbin/udevd|'                                          \
    /usr/share/man/man8/systemd-udevd.service.8                         \
  > /usr/share/man/man8/udevd.8

rm /usr/share/man/man*/systemd*
```

Enfin, déconfigurez la variable `udev_helpers` :

```
unset udev_helpers
```

8.76.2. Configurer Udev

Les informations sur les périphériques matériels sont maintenues dans les répertoires `/etc/udev/hwdb.d` et `/usr/lib/udev/hwdb.d`. Udev a besoin que ces informations soient compilées en une base de données binaire `/etc/udev/hwdb.bin`. Créez la base de données initiale :

```
udev-hwdb update
```

Cette commande doit être lancée à chaque fois que les informations sur le matériel sont mises à jour.

8.76.3. Contenu d'Udev

Programmes installés: udevadm, udevd (lien symbolique vers udevadm) et udev-hwdb
Bibliothèques installées: libudev.so
Répertoires installés: /etc/udev et /usr/lib/udev

Descriptions courtes

udevadm Outil d'administration générique d'udev : contrôle le démon udevd, fournit des informations sur la base de données Udev, suit les uevents, attend la fin de certains uevents, teste la configuration d'Udev et génère des uevents pour un périphérique donné

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

udev-hwdb Met à jour ou exécute des requêtes sur la base de données du matériel.

`libudev` Une interface de bibliothèque aux informations de périphériques d'udev

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

8.77. Man-DB-2.13.1

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

Temps de construction approximatif: 0,3 SBU
Espace disque requis: 44 Mo

8.77.1. Installation de Man-DB

Préparez la compilation de man-DB :

```
./configure --prefix=/usr                         \
            --docdir=/usr/share/doc/man-db-2.13.1 \
            --sysconfdir=/etc                     \
            --disable-setuid                      \
            --enable-cache-owner=bin              \
            --with-browser=/usr/bin/lynx          \
            --with-vgrind=/usr/bin/vgrind         \
            --with-grap=/usr/bin/grap             \
            --with-systemdtmpfilesdir=            \
            --with-systemdsystemunitdir=
```

Voici la signification des options de configuration :

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

`--enable-cache-owner=bin`
Cela attribue les fichiers de cache du système à l'utilisateur `bin`.

`--with-...`
Ces trois paramètres sont utilisés pour initialiser quelques programmes par défaut. **lynx** est un navigateur Web en mode console (voir BLFS pour les instructions d'installation), **vgrind** convertit du code source de programme en entrée Groff et **grap** est utile pour la composition de texte de graphes dans les documents Groff. Les programmes **vgrind** et **grap** ne sont normalement pas nécessaires pour la visualisation des pages de manuel. Ils ne font pas partie de LFS ou de BLFS, mais vous devriez pouvoir les installer vous-même après avoir fini LFS si vous le souhaitez.

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

Compilez le paquet :

```
make
```

Pour tester les résultats, exécutez :

```
make check
```

Installez le paquet :

```
make install
```

8.77.2. Pages de manuel non anglophones dans LFS

Le tableau suivant montre l'encodage présumé avec lequel Man-DB encodera les pages de manuel installées dans `/usr/share/man/<ll>`. En outre, Man-DB détermine correctement si les pages de manuel installées dans ce répertoire sont encodées en UTF-8.

Tableau 8.1. Encodage de caractère attendu des pages de manuel 8-bit de base

Langue (code)	Encodage	Langue (code)	Encodage
Danois (da)	ISO-8859-1	Croate (hr)	ISO-8859-2
Allemand (de)	ISO-8859-1	Hongrois (hu)	ISO-8859-2
Anglais (en)	ISO-8859-1	Japonais (ja)	EUC-JP
Espagnol (es)	ISO-8859-1	Coréen (ko)	EUC-KR
Estonien (et)	ISO-8859-1	Lituanien (lt)	ISO-8859-13
Finnois (fi)	ISO-8859-1	Letton (lv)	ISO-8859-13
Français (fr)	ISO-8859-1	Macédonien (mk)	ISO-8859-5
Irlandais (ga)	ISO-8859-1	Polonais (pl)	ISO-8859-2
Galicien (gl)	ISO-8859-1	Roumain (ro)	ISO-8859-2
Indonésien (id)	ISO-8859-1	Grec (el)	ISO-8859-7
Islandais (is)	ISO-8859-1	Slovaque (sk)	ISO-8859-2
Italien (it)	ISO-8859-1	Slovène (sl)	ISO-8859-2
Norvégien bokmål (nb)	ISO-8859-1	Latin serbe (sr@latin)	ISO-8859-2
Hollandais (nl)	ISO-8859-1	Serbe (sr)	ISO-8859-5
Norvégien Nynorsk (nn)	ISO-8859-1	Turc (tr)	ISO-8859-9
Norvégien (no)	ISO-8859-1	Ukrainien (uk)	KOI8-U
Portugais (pt)	ISO-8859-1	Vietnamien (vi)	TCVN5712-1
Suédois (sv)	ISO-8859-1	Chinois simplifié (zh_CN)	GBK
Biélorusse (be)	CP1251	Chinois simplifié, Singapour (zh_SG)	GBK
Bulgare (bg)	CP1251	Chinois traditionnel, Hong Kong (zh_HK)	BIG5HKSCS
Tchèque (cs)	ISO-8859-2	Chinois traditionnel (zh_TW)	BIG5

Note

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

8.77.3. Contenu de Man-DB

Programmes installés: accessdb, apropos (lien vers whatis), catman, lexgrog, man, man-recode, mandb, manpath et whatis
Bibliothèques installées: libman.so et libmandb.so (tous deux dans /usr/lib/man-db)
Répertoires installés: /usr/lib/man-db, /usr/libexec/man-db et /usr/share/doc/man-db-2.13.1

Descriptions courtes

accessdb Transforme le contenu de la base de données **whatis** en format lisible par un humain

apropos Recherche la base de données **whatis** et affiche les descriptions courtes des commandes système qui contiennent une chaîne donnée

catman	Crée ou met à jour les pages de manuel préformatées
lexgrog	Affiche des informations sous forme de résumé d'une ligne à propos d'une page de manuel donnée
man	Formate et affiche les pages de manuel demandées
man-recode	Converti les pages de manuel vers un autre encodage
mandb	Crée ou met à jour la base de données **whatis**
manpath	Affiche le contenu de $MANPATH ou (si $MANPATH n'est pas paramétré) d'un chemin de recherche convenable basé sur les paramètres de man.conf et de l'environnement de l'utilisateur
whatis	Recherche la base de données **whatis** et affiche les descriptions courtes des commandes système qui contiennent le mot-clé donné sous forme d'un mot séparé
`libman`	Contient le support au moment de l'exécution de **man**
`libmandb`	Contient le support au moment de l'exécution de **man**

8.78. Procps-ng-4.0.5

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

Temps de construction approximatif: 0,1 SBU
Espace disque requis: 28 Mo

8.78.1. Installation de Procps-ng

Préparez maintenant la compilation de procps-ng :

```
./configure --prefix=/usr                           \
            --docdir=/usr/share/doc/procps-ng-4.0.5 \
            --disable-static                        \
            --disable-kill                          \
            --enable-watch8bit
```

Voici la signification de l'option de configuration :

`--disable-kill`
Cette option désactive la construction de la commande **kill** installée dans le paquet util-linux.

`--enable-watch8bit`
Ce paramètre active la prise en charge de ncursesw dans la commande **watch**, pour qu'elle puisse gérer les caractères sur 8 bits.

Compilez le paquet :

```
make
```

Pour lancer la suite de tests, exécutez :

```
chown -R tester .
su tester -c "PATH=$PATH make check"
```

Un test nommé `ps with output flag bsdtime,cputime,etime,etimes` est connu pour échouer si le noyau hôte n'est pas construit avec l'option `CONFIG_BSD_PROCESS_ACCT` activée. De plus, un test de pgrep peut échouer dans l'environnement chroot.

Installez le paquet :

```
make install
```

8.78.2. Contenu de Procps-ng

Programmes installés: free, pgrep, pidof, pkill, pmap, ps, pwdx, slabtop, sysctl, tload, top, uptime, vmstat, w et watch
Bibliothèque installée: libproc-2.so
Répertoires installés: /usr/include/procps et /usr/share/doc/procps-ng-4.0.5

Descriptions courtes

free Indique le total de mémoire libre et utilisé sur le système à la fois pour la mémoire physique et pour la mémoire swap

pgrep Recherche les processus suivant leur nom et autres attributs

pidof Indique le PID des programmes précisés

pkill Envoie des signaux aux processus suivant leur nom et autres attributs

pmap Affiche le plan mémoire du processus désigné

ps	Donne un aperçu des processus en cours d'exécution
pwdx	Indique le répertoire d'exécution courant d'un processus
slabtop	Affiche des informations détaillées sur le cache slab du noyau en temps réel
sysctl	Modifie les paramètres du noyau en cours d'exécution
tload	Affiche un graphe de la charge système actuelle
top	Affiche une liste des processus demandant le maximum de ressources CPU ; il fournit un affichage agréable sur l'activité du processeur en temps réel
uptime	Affiche le temps d'exécution du système, le nombre d'utilisateurs connectés et les moyennes de charge système
vmstat	Affiche les statistiques de mémoire virtuelle, donne des informations sur les processus, la mémoire, la pagination, le nombre de blocs en entrées/sorties, les échappements et l'activité CPU
w	Affiche les utilisateurs actuellement connectés, où et depuis quand
watch	Lance une commande de manière répétée, affichant le premier écran de sa sortie ; ceci vous permet de surveiller la sortie
`libproc-2`	Contient les fonctions utilisées par la plupart des programmes de ce paquet

8.79. Util-linux-2.41.1

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

Temps de construction approximatif: 0,5 SBU
Espace disque requis: 346 Mo

8.79.1. Installation d'Util-linux

Préparez la compilation d'Util-linux :

```
./configure --bindir=/usr/bin     \
            --libdir=/usr/lib     \
            --runstatedir=/run    \
            --sbindir=/usr/sbin   \
            --disable-chfn-chsh   \
            --disable-login       \
            --disable-nologin     \
            --disable-su          \
            --disable-setpriv     \
            --disable-runuser     \
            --disable-pylibmount  \
            --disable-liblastlog2 \
            --disable-static      \
            --without-python      \
            --without-systemd     \
            --without-systemdsystemunitdir        \
            ADJTIME_PATH=/var/lib/hwclock/adjtime \
            --docdir=/usr/share/doc/util-linux-2.41.1
```

Les options --disable et --without préviennent des avertissements à propos d'éléments de construction qui requièrent des paquets non compris dans LFS ou incohérents avec les programmes installés par d'autres paquets.

Compilez le paquet :

```
make
```

Si vous le souhaitez, créez un fichier `/etc/fstab` factice pour satisfaire deux tests et exécutez la suite de tests en tant qu'utilisateur non-`root` :

Avertissement

L'exécution de la suite de tests en tant qu'utilisateur `root` peut être dangereuse pour votre système. Pour la lancer, l'option CONFIG_SCSI_DEBUG du noyau doit être disponible sur le système en cours d'exécution et doit être construite en tant que module. Si elle est compilée en dur dans le noyau, cela empêchera le démarrage. Pour une exécution complète, il faut installer d'autres paquets de BLFS. Si vous le souhaitez, vous pouvez lancer ce test après le redémarrage dans le système LFS terminé, en exécutant :

```
bash tests/run.sh --srcdir=$PWD --builddir=$PWD
```

```
touch /etc/fstab
chown -R tester .
su tester -c "make -k check"
```

Les tests *hardlink* échoueront si le noyau de l'hôte n'a pas l'option `CONFIG_CRYPTO_USER_API_HASH` ou n'a aucune des options qui fournissent une implémentation de SHA256 (par exemple, `CONFIG_CRYPTO_SHA256` ou `CONFIG_CRYPTO_SHA256_SSSE3` si le CPU prend les suppléments SSE3 en charge). En plus, le test lsfd: inotify échouera si l'option du noyau `CONFIG_NETLINK_DIAG` n'est pas activée.

Un test, `kill: decode functions`, est connu pour échouer avec bash5.3-rc1 ou supérieur.

Installez le paquet :

```
make install
```

8.79.2. Contenu d'Util-linux

Programmes installés: addpart, agetty, blkdiscard, blkid, blkzone, blockdev, cal, cfdisk, chcpu, chmem, choom, chrt, col, colcrt, colrm, column, ctrlaltdel, delpart, dmesg, eject, fallocate, fdisk, fincore, findfs, findmnt, flock, fsck, fsck.cramfs, fsck.minix, fsfreeze, fstrim, getopt, hexdump, hwclock, i386, ionice, ipcmk, ipcrm, ipcs, irqtop, isosize, kill, last, lastb (lien vers last), ldattach, linux32, linux64, logger, look, losetup, lsblk, lscpu, lsipc, lsirq, lslocks, lslogins, lsmem, lsns, mcookie, mesg, mkfs, mkfs.bfs, mkfs.cramfs, mkfs.minix, mkswap, more, mount, mountpoint, namei, nsenter, partx, pivot_root, prlimit, readprofile, rename, renice, resizepart, rev, rfkill, rtcwake, script, scriptlive, scriptreplay, setarch, setsid, setterm, sfdisk, sulogin, swaplabel, swapoff (lien vers swapon), swapon, switch_root, taskset, uclampset, ul, umount, uname26, unshare, utmpdump, uuidd, uuidgen, uuidparse, wall, wdctl, whereis, wipefs, x86_64 et zramctl

Bibliothèques installées: libblkid.so, libfdisk.so, libmount.so, libsmartcols.so et libuuid.so

Répertoires installés: /usr/include/blkid, /usr/include/libfdisk, /usr/include/libmount, /usr/include/libsmartcols, /usr/include/uuid, /usr/share/doc/util-linux-2.41.1 et /var/lib/hwclock

Descriptions courtes

addpart Informe le noyau Linux de nouvelles partitions

agetty Ouvre un port tty, demande un nom de connexion puis appelle le programme **login**

blkdiscard Désactive des secteurs d'un périphérique

blkid Un outil en ligne de commande pour trouver et afficher les attributs d'un périphérique bloc

blkzone Lance des commandes de zone sur le périphérique bloc donné

blockdev Permet aux utilisateurs d'appeler les ioctl d'un périphérique bloc à partir de la ligne de commande

cal Affiche un calendrier simple

cfdisk Manipule la table des partitions du périphérique donné

chcpu Modifie l'état des processeurs

chmem Configure la mémoire

choom Affiche et adapte les résultats OOM-killer qui déterminent quel processus supprimer en premier quand la mémoire de Linux sature

chrt Manipule les attributs d'un processus en temps réel

col Filtre les retours de chariot inversés

colcrt Filtre la sortie de **nroff** pour les terminaux manquant de capacités comme le texte barré ou les demi-lignes

colrm Filtre les colonnes données

column Formate un fichier donné en plusieurs colonnes

ctrlaltdel Initialise la combinaison des touches Ctrl+Alt+Del pour une réinitialisation matérielle ou logicielle

delpart Demande au noyau Linux de supprimer une partition

dmesg Affiche les messages du noyau lors du démarrage

eject Éjecte un média amovible

fallocate Pré-alloue de l'espace à un fichier

fdisk Manipule la table des partitions du périphérique donné

fincore Compte les pages de contenu d'un fichier en mémoire

findfs Trouve un système de fichiers par étiquette ou UUID (*Universally Unique Identifier*, soit Identifiant Unique Universel)

findmnt Est une interface en ligne de commande avec la bibliothèque libmount pour du travail avec les fichiers mountinfo, fstab et mtab

flock Acquiert le verrouillage d'un fichier puis exécute une commande en maintenant le verrouillage

fsck Est utilisé pour vérifier, et parfois réparer, les systèmes de fichiers

fsck.cramfs Réalise un test de cohérence sur le système de fichiers Cramfs du périphérique donné

fsck.minix Réalise un test de cohérence sur le système de fichiers Minix du périphérique donné

fsfreeze Est une enveloppe très simple autour des opérations du pilote noyau FIFREEZE/FITHAW ioctl

fstrim Écarte les blocs inutilisés sur un système de fichiers monté

getopt Analyse les options sur la ligne de commande donnée

hardlink Crée des liens durs pour renforcer les fichiers dupliqués

hexdump Affiche le fichier indiqué en hexadécimal ou dans un autre format donné

hwclock Lit ou initialise l'horloge du matériel, aussi appelée horloge RTC (*Real-Time Clock*, horloge à temps réel) ou horloge du BIOS (*Basic Input-Output System*)

i386 Un lien symbolique vers setarch

ionice Obtient ou initialise la classe de planification IO (ES) et la priorité pour un programme

ipcmk Crée diverses ressources IPC

ipcrm Supprime la ressource IPC (inter-process communication) donnée

ipcs Fournit l'information de statut IPC

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

isosize Affiche la taille d'un système de fichiers iso9660

kill Envoie des signaux aux processus

last Affiche les utilisateurs connectés (et déconnectés) dernièrement en s'appuyant sur le fichier `/var/log/wtmp` ; il affiche également les démarrages du système, les extinctions et les changements de niveau d'exécution

lastb Affiche les tentatives de connexions enregistrées dans `/var/log/btmp`

ldattach Attache une discipline de ligne à une ligne série

linux32 Un lien symbolique vers setarch

linux64 Un lien symbolique vers setarch

logger Enregistre le message donné dans les traces système

look Affiche les lignes commençant par la chaîne donnée

losetup Initialise et contrôle les périphériques loop

lsblk Liste les informations sur tous les périphériques blocs ou ceux sélectionnés dans un format semblable à une arborescence

lscpu Affiche des informations sur l'architecture du processeur

lsfd	Montre les informations sur les fichiers ouverts ; remplace **lsof**
lsipc	Affiche les informations sur les fonctions IPC actuellement utilisées sur le système
lsirq	Affiche les information du compteur d'interruption du noyau
lslocks	Liste les verrous du système local
lslogins	Liste les informations sur les comptes utilisateurs, groupes et systèmes
lsmem	Liste les intervalles de mémoire disponibles avec leur statut en ligne
lsns	Liste les espaces de noms
mcookie	Génère des cookies magiques, nombres hexadécimaux aléatoires sur 128 bits, pour **xauth**
mesg	Contrôle si d'autres utilisateurs peuvent envoyer des messages au terminal de l'utilisateur actuel
mkfs	Construit un système de fichiers sur un périphérique (habituellement une partition du disque dur)
mkfs.bfs	Crée un système de fichiers bfs de SCO (Santa Cruz Operations)
mkfs.cramfs	Crée un système de fichiers cramfs
mkfs.minix	Crée un système de fichiers Minix
mkswap	Initialise le périphérique ou le fichier à utiliser comme swap
more	Est un filtre pour visualiser un texte un écran à la fois
mount	Attache le système de fichiers du périphérique donné sur un répertoire spécifié dans le système de fichiers
mountpoint	Vérifie si le répertoire est un point de montage
namei	Affiche les liens symboliques dans les chemins donnés
nsenter	Lance un programme avec un nom espacé des autres processus
partx	Signale au noyau la présence et le nombre de partitions sur un disque
pivot_root	Fait en sorte que le système de fichiers donné soit le nouveau système de fichiers racine du processus actuel
prlimit	Récupère et envoie la limite des ressources d'un processus
readprofile	Lit les informations de profilage du noyau
rename	Renomme les fichiers donnés, remplaçant une chaîne donnée par une autre
renice	Modifie la priorité des processus exécutés
resizepart	Demande au noyau Linux de redimensionner une partition
rev	Inverse les lignes d'un fichier donné
rfkill	Outil pour activer et désactiver les périphériques sans fil
rtcwake	Utilisé pour mettre un système en sommeil jusqu'à un moment de réveil spécifié
script	Crée un script type à partir d'une session du terminal
scriptlive	Rejoue des scripts type de session en utilisant les informations de temps
scriptreplay	Rejoue des scripts type en utilisant les informations de temps
setarch	Change d'architecture signalée dans un nouvel environnement de programme et initialise les commutateurs adéquats
setsid	Lance le programme donné dans une nouvelle session
setterm	Initialise les attributs du terminal

sfdisk Est un manipulateur de table de partitions disque

sulogin Permet la connexion de `root`. Il est normalement appelé par **init** lorsque le système passe en mono-utilisateur

swaplabel Permet de modifier l'UUID et l'étiquette d'un espace d'échange

swapoff Désactive des périphériques et des fichiers pour la pagination et l'échange

swapon Active les périphériques et fichiers de pagination et d'échange et liste les périphériques et fichiers en cours d'utilisation

switch_root Change de système de fichiers racine pour une arborescence montée

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

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

ul Un filtre pour traduire les soulignements en séquences d'échappement indiquant un soulignement pour le terminal utilisé

umount Déconnecte un système de fichiers à partir de la hiérarchie de fichiers du système

uname26 Un lien symbolique vers setarch

unshare Lance un programme avec quelques espaces de nom non partagés avec le parent

utmpdump Affiche le contenu du fichier de connexion donné dans un format convivial

uuidd Un démon utilisé par la bibliothèque UUID pour générer des UUIDs basés sur l'heure de manière sécurisée et avec une garantie d'unicité

uuidgen Crée un nouvel UUID. Chaque nouvel UUID peut être raisonnablement considéré unique parmi tous les UUID créés, sur le système local mais aussi sur les autres, dans le passé et dans le futur, avec une forte probabilité (2^{128} UUIDS sont possibles)

uuidparse Un utilitaire pour analyser des identifiants uniques

wall Affiche le contenu d'un fichier ou, par défaut, son entrée standard, sur les terminaux de tous les utilisateurs actuellement connectés

wdctl Affiche l'état du watchdog matériel

whereis Affiche l'emplacement du binaire, les sources et la page de manuel de la commande donnée

wipefs Nettoie la signature d'un système de fichiers à partir du périphérique

x86_64 Un lien symbolique vers setarch

zramctl Un programme pour configurer et contrôler les périphériques zram (mémoire ram compressée)

`libblkid` Contient des routines pour l'identification des périphériques et l'extraction des modèles

`libfdisk` Contient des routines pour la manipulation de table de partition

`libmount` Contient les routines pour le montage et le démontage des périphériques de bloc

`libsmartcols` Contient les routines pour la sortie d'écran d'aide sous forme de tableau

`libuuid` Contient des routines pour la génération d'identifiants uniques pour des objets qui peuvent être accessibles en dehors du système local

8.80. E2fsprogs-1.47.3

Le paquet e2fsprogs contient les outils de gestion du système de fichiers `ext2`. Il prend aussi en charge les systèmes de fichiers journalisés `ext3` et `ext4`.

Temps de construction approximatif: 2,4 SBU sur un disque dur, 0,5 SBU sur un SSD
Espace disque requis: 100 Mo

8.80.1. Installation de E2fsprogs

La documentation d'e2fsprogs recommande de construire le paquet dans un sous-répertoire du répertoire source :

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

Préparez la compilation d'e2fsprogs :

```
../configure --prefix=/usr       \
             --sysconfdir=/etc   \
             --enable-elf-shlibs \
             --disable-libblkid  \
             --disable-libuuid   \
             --disable-uuidd     \
             --disable-fsck
```

Voici la signification des options de configuration :

`--enable-elf-shlibs`
Cette option crée les bibliothèques partagées que certains programmes de ce paquet utilisent.

`--disable-*`
Cette option empêche la construction et l'installation des bibliothèques `libuuid` et `libblkid`, du démon `uuidd` et de l'outil **fsck**, car util-Linux installe des versions plus récentes.

Compilez le paquet :

```
make
```

Pour lancer les tests, lancez :

```
make check
```

Un test nommé `m_assume_storage_prezeroed` est connu pour échouer. Un autre test nommé `m_rootdir_acl` est connu pour échouer si le système de fichier utilisé pour le système LFS n'est pas de type `ext4`.

Installez le paquet :

```
make install
```

Supprimez les bibliothèques statiques inutiles :

```
rm -fv /usr/lib/{libcom_err,libe2p,libext2fs,libss}.a
```

Ce paquet installe un fichier `.info` gzippé mais ne met pas à jour le fichier `dir` du système. Dézippez ce fichier puis mettez à jour le fichier `dir` du système en utilisant les commandes suivantes :

```
gunzip -v /usr/share/info/libext2fs.info.gz
install-info --dir-file=/usr/share/info/dir /usr/share/info/libext2fs.info
```

Si vous le désirez, créez et installez de la documentation supplémentaire en exécutant les commandes suivantes :

```
makeinfo -o      doc/com_err.info ../lib/et/com_err.texinfo
install -v -m644 doc/com_err.info /usr/share/info
install-info --dir-file=/usr/share/info/dir /usr/share/info/com_err.info
```

8.80.2. Configuration de E2fsprogs

`/etc/mke2fs.conf` contient les valeurs par défaut de diverses optionsqde la ligne de commande de **mke2fs**. Vous pouvez modifier le fichier pour rendre les valeurs par défaut plus utiles. Par exemple, certains utilitaires (absents de LFS et BLFS) ne peuvent pas reconnaitre un système de fichiers `ext4` avec la fonctionnalité `metadata_csum_seed`. **Si** vous avez besoin d'un tel utilitaire, vous pouvez supprimer la fonctionnalité de la liste des fonctionnalités par défaut pour `ext4` avec la commande :

```
sed 's/metadata_csum_seed,//' -i /etc/mke2fs.conf
```

Consultez la page de manuel *mke2fs.conf(5)* pour plus de détails.

8.80.3. Contenu de E2fsprogs

Programmes installés:	badblocks, chattr, compile_et, debugfs, dumpe2fs, e2freefrag, e2fsck, e2image, e2label, e2mmpstatus, e2scrub, e2scrub_all, e2undo, e4crypt, e4defrag, filefrag, fsck.ext2, fsck.ext3, fsck.ext4, logsave, lsattr, mk_cmds, mke2fs, mkfs.ext2, mkfs.ext3, mkfs.ext4, mklost+found, resize2fs et tune2fs
Bibliothèques installées:	libcom_err.so, libe2p.so, libext2fs.so et libss.so
Répertoires installés:	/usr/include/e2p, /usr/include/et, /usr/include/ext2fs, /usr/include/ss, /usr/lib/e2fsprogs, /usr/share/et et /usr/share/ss

Descriptions courtes

badblocks	Recherche les blocs défectueux sur un périphérique (habituellement une partition d'un disque)
chattr	Modifie les propriétés des fichiers sur les systèmes de fichiers `ext{234}`
compile_et	Un compilateur de table d'erreurs qui convertit une table de noms d'erreurs et des messages associés en un fichier source C à utiliser avec la bibliothèque `com_err`
debugfs	Un débogueur de système de fichiers qui peut être utilisé pour analyser et modifier l'état de systèmes de fichiers `ext{234}`
dumpe2fs	Affiche le superbloc et les informations de groupes de blocs sur le système de fichiers présent sur un périphérique donné
e2freefrag	Rend compte des informations de fragmentation de l'espace libre
e2fsck	Permet de vérifier et le cas échéant, de réparer des systèmes de fichiers `ext{234}`
e2image	Permet de sauvegarder les données critiques d'un système de fichiers `ext{234}` dans un fichier
e2label	Affiche ou modifie le label d'un système de fichiers `ext{234}` sur un périphérique donné
e2mmpstatus	Vérifie le statut MMP d'un système de fichiers `ext4`
e2scrub	Vérifie le contenu d'un système de fichiers monté `ext{234}`
e2scrub_all	Recherche les erreurs des systèmes de fichiers montés `ext{234}`
e2undo	Rejoue le log d'annulation pour un système de fichiers `ext{234}` trouvé sur un périphérique. Il peut être utilisé pour annuler une opération échouée par un programme e2fsprogs.
e4crypt	Utilitaire de chiffrement du système de fichiers `ext4`
e4defrag	Défragmenteur en ligne pour les systèmes de fichiers `ext4`
filefrag	Signale à quel point un fichier particulier peut être mal fragmenté
fsck.ext2	Vérifie par défaut les systèmes de fichiers `ext2`. C'est un lien matériel vers **e2fsck**
fsck.ext3	Vérifie par défaut les systèmes de fichiers `ext3`. C'est un lien matériel vers **e2fsck**
fsck.ext4	Vérifie par défaut les systèmes de fichiers `ext4`. C'est un lien matériel vers **e2fsck**

logsave	Sauvegarde la sortie d'une commande dans un fichier journal
lsattr	Liste les attributs de fichiers sur un système de fichiers ext2
mk_cmds	Convertit une table de noms de commandes et de messages d'aide en un fichier source C utilisable avec la bibliothèque sous-système `libss`
mke2fs	Crée un système de fichiers `ext{234}` sur le périphérique spécifié
mkfs.ext2	Crée par défaut un système de fichiers `ext2`. C'est un lien matériel vers **mke2fs**
mkfs.ext3	Crée par défaut un système de fichiers `ext3`. C'est un lien matériel vers **mke2fs**
mkfs.ext4	Crée par défaut un système de fichiers `ext4`. C'est un lien matériel vers **mke2fs**
mklost+found	Crée un répertoire `lost+found` sur un système de fichiers `ext{234}`. Il pré-alloue des blocs de disque à ce répertoire pour alléger la tâche d'**e2fsck**
resize2fs	Permet d'agrandir ou de réduire des systèmes de fichiers `ext{234}`
tune2fs	Ajuste les paramètres des systèmes de fichiers `ext{234}`
`libcom_err`	La routine d'affichage d'erreurs courantes
`libe2p`	Est utilisé par **dumpe2fs**, **chattr** et **lsattr**
`libext2fs`	Contient des routines pour permettre aux programmes de niveau utilisateur de manipuler des systèmes de fichiers `ext{234}`
`libss`	Est utilisé par **debugfs**

8.81. Sysklogd-2.7.2

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

Temps de construction approximatif: moins de 0,1 SBU
Espace disque requis: 3,9 Mo

8.81.1. Installation de Sysklogd

Préparez le paquet pour la compilation :

```
./configure --prefix=/usr      \
            --sysconfdir=/etc  \
            --runstatedir=/run \
            --without-logger   \
            --disable-static   \
            --docdir=/usr/share/doc/sysklogd-2.7.2
```

Compilez le paquet :

```
make
```

Ce paquet n'a pas de suite de tests.

Installez le paquet :

```
make install
```

8.81.2. Configuration de Sysklogd

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

```
cat > /etc/syslog.conf << "EOF"
# Début de /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 *

# Ne pas ouvrir de ports internet.
secure_mode 2

# Fin de /etc/syslog.conf
EOF
```

8.81.3. Contenu de Sysklogd

Programme installé: syslogd

Descriptions courtes

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

8.82. SysVinit-3.14

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

Temps de construction approximatif: moins de 0,1 SBU
Espace disque requis: 3,9 Mo

8.82.1. Installation de SysVinit

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

```
patch -Np1 -i ../sysvinit-3.14-consolidated-1.patch
```

Compilez le paquet :

```
make
```

Ce paquet n'a pas de suite de tests.

Installez le paquet :

```
make install
```

8.82.2. Contenu de SysVinit

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

Descriptions courtes

bootlogd Trace les messages de démarrage dans le journal

fstab-decode Lance une commande avec les arguments fstab-encoded

halt Lance le programme **shutdown** avec l'option *-h*. S'il est déjà au niveau d'exécution 0, il demande au noyau d'arrêter le système et il note dans le fichier `/var/log/wtmp` que le système est en cours d'arrêt

init Le premier processus à exécuter lorsque le noyau a initialisé le matériel. Ce processus prend la main sur le démarrage et lance les processus indiqués dans son fichier de configuration

killall5 Envoie un signal à tous les processus en excluant ceux de sa propre session afin de ne pas arrêter le processus parent

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

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

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

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

telinit Dit à **init** quel niveau d'exécution adopter

8.83. À propos des symboles de débogage

La plupart des programmes et des bibliothèques sont compilés, par défaut, en incluant les symboles de débogage (avec l'option *-g* de **gcc**). Ceci signifie que, lors du débogage d'un programme ou d'une bibliothèque compilés avec les informations de débogage, le débogueur peut vous donner non seulement les adresses mémoire mais aussi le nom des routines et des variables.

Néanmoins, l'intégration de ces symboles de débogage font grossir le programme ou la bibliothèque de façon significative. Voici des exemples de l'espace occupé par ces symboles :

- Un binaire **bash** avec les symboles de débogage : 1 200 Ko
- Un binaire **bash** sans les symboles de débogage : 480 Ko
- Les fichiers Glibc et GCC (`/lib` et `/usr/lib`) avec les symboles de débogage : 87 Mo
- Les fichiers Glibc et GCC sans les symboles de débogage : 16 Mo

La taille des fichiers va varier en fonction de quel compilateur ou de quelle bibliothèque du C a été utilisé. Cependant, un programme dont les symboles de débogage ont été supprimés est en général plus léger de 50 à 80% comparé à un programme qui les a encore. Comme la plupart des gens n'utiliseront jamais un débogueur sur leur système, beaucoup d'espace disque peut être gagné en supprimant ces symboles. La prochaine section montre comment supprimer tous les symboles de débogage des programmes et bibliothèques.

8.84. Nettoyage

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

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

La commande **strip** avec le paramètre *–strip-unneeded* supprime tous les symboles de débogage d'un binaire ou d'une bibliothèque. Elle supprime aussi toutes les entrées de la table des symboles qui ne sont normalement pas requises par l'éditeur de liens (pour les binaires liés dynamiquement et les bibliothèques partagées). Utiliser *--strip-debug* ne supprime pas les entrées de la table des symboles qui peuvent être requises par certaines applications. La différence entre `unneeded` et `debug` est très faible. Par exemple une `libc.a` non nettoyée pèse 22,4 Mo. Après le nettoyage avec *--strip-debug* elle pèse 5,9 Mo. Utiliser *--strip-unneeded* ne réduit sa taille qu'à 5,8 Mo.

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

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

Note

Le nom du chargeur ELF est ld-linux-x86-64.so.2 sur les systèmes 64 bits et ld-linux.so.2 sur les systèmes 32 bits. La construction ci-dessous choisit le bon nom pour l'architecture actuelle, en excluant tout ce qui fini en `g`, au cas où les commandes ci-dessous ont déjà été lancées.

Important

S'il y a un paquet dont la version est différente de la version spécifiée dans le livre (que se soit pour suivre une recommendation de sécurité ou pour vos préférences personnelles), il peut être nécessaire de mettre à jour le nom de fichier de la bibliothèque dans `save_usrlib` ou `online_usrlib`. **Si vous ne le faites pas, votre système peut devenir complètement inutilisable.**

```
save_usrlib="$(cd /usr/lib; ls ld-linux*[^g])
             libc.so.6
             libthread_db.so.1
             libquadmath.so.0.0.0
             libstdc++.so.6.0.34
             libitm.so.1.0.0
             libatomic.so.1.2.0"

cd /usr/lib

for LIB in $save_usrlib; do
    objcopy --only-keep-debug --compress-debug-sections=zstd $LIB $LIB.dbg
    cp $LIB /tmp/$LIB
    strip --strip-debug /tmp/$LIB
    objcopy --add-gnu-debuglink=$LIB.dbg /tmp/$LIB
    install -vm755 /tmp/$LIB /usr/lib
    rm /tmp/$LIB
done

online_usrbin="bash find strip"
online_usrlib="libbfd-2.45.so
               libsframe.so.2.0.0
               libhistory.so.8.3
               libncursesw.so.6.5
               libm.so.6
               libreadline.so.8.3
               libz.so.1.3.1
               libzstd.so.1.5.7
               $(cd /usr/lib; find libnss*.so* -type f)"

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

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

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

unset BIN LIB save_usrlib online_usrbin online_usrlib
```

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

8.85. Nettoyage

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

```
rm -rf /tmp/{*,.*}
```

Il y a aussi plusieurs fichiers installés dans les répertoires /usr/lib et /usr/libexec avec l'extension .la. Ce sont des fichiers « d'archive libtool ». Sur les systèmes Linux modernes, ils ne sont utiles que pour libltdl. Aucune bibliothèque de LFS ne devrait être chargée par libltdl et certains fichiers .la sont connus pour empêcher la construction des paquets de BLFS. Supprimez maintenant ces fichiers :

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

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

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

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

Enfin, supprimez le compte utilisateur Â« tester Â» crÃ©Ã© au dÃ©but du chapitre prÃ©cÃ©dent.

```
userdel -r tester
```

Chapitre 9. Configuration du système

9.1. Introduction

Le démarrage d'un système Linux englobe plusieurs tâches. Le processus implique de monter les systèmes de fichiers virtuels et réels, d'initialiser les périphériques, de vérifier la conformité des systèmes de fichier, de monter et activer tout fichier d'échange ou partition d'échange, de régler l'horloge du système, d'activer le réseau, de démarrer les démons nécessaires au système et d'accomplir d'autres tâches personnalisées précisées par l'utilisateur. Il faut organiser ce processus afin de s'assurer que les tâches s'effectuent dans l'ordre et le plus vite possible.

9.1.1. System V

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

Le programme **init** est contrôlé par le fichier `/etc/inittab` et s'organise en niveaux d'exécution que l'utilisateur peut choisir. Dans LFS ces niveaux s'utilisent ainsi :

```
0 — arrêt
1 — Mode mono-utilisateur
2 — Définissable par l'utilisateur
3 — Mode multi-utilisateur complet
4 — Définissable par l'utilisateur
5 — Mode multi-utilisateur complet avec gestionnaire d'affichage
6 — redémarrage
```

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

Avantages

- Système stabilisé et maîtrisé.
- Facile à personnaliser.

Inconvénients

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

9.2. LFS-Bootscripts-20250827

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

Temps de construction approximatif: moins de 0,1 SBU
Espace disque requis: 238 Ko

9.2.1. Installation de LFS-Bootscripts

Installez le paquet :

```
make install
```

9.2.2. Contenu de LFS-Bootscripts

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

Descriptions courtes

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

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

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

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

halt Arrête le système

ifdown Arrête un périphérique réseau

ifup Initialise un périphérique réseau

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

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

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

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

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

rc Script de contrôle du niveau d'exécution maître ; il est responsable du lancement individuel des autres scripts de démarrage, selon une séquence déterminée par les noms des liens symboliques référençant les scripts de démarrage concernés

reboot Redémarre le système

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

setclock Réinitialise l'horloge du système avec l'heure locale au cas où l'horloge matérielle ne soit pas réglé en UTC

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

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

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

sysklogd Lance et arrête les démons des journaux système et noyau

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

udev Prépare le répertoire `/dev` et démarre le daemon d'udev

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

9.3. Manipulation des périphériques et modules

Au Chapitre 8, nous avons installé le paquet udev en construisant udev . Avant d'entrer dans les détails concernant son fonctionnement, un bref historique des méthodes précédentes de gestion des périphériques est nécessaire.

Traditionnellement, les systèmes Linux utilisaient une méthode de création de périphériques statiques avec laquelle un grand nombre de nœuds de périphériques étaient créés sous `/dev` (quelques fois littéralement des milliers de nœuds), que le matériel correspondant existe ou non. Ceci était le plus souvent réalisé avec un script **MAKEDEV**, qui contient des appels au programme **mknod** avec les numéros de périphériques majeurs et mineurs pour chaque périphérique possible qui pourrait exister dans le monde.

En utilisant la méthode udev, seuls les nœuds des périphériques détectés par le noyau sont créés. Comme ces nœuds de périphériques sont créés à chaque lancement du système, ils sont stockés dans un système de fichiers `devtmpfs` (un système de fichiers virtuel qui réside entièrement dans la mémoire du système). Les nœuds de périphériques ne requièrent pas beaucoup d'espace, donc la mémoire utilisée est négligeable.

9.3.1. Historique

En février 2000, un nouveau système de fichiers appelé `devfs` a été intégré au noyau 2.3.46 et rendu disponible pour la série 2.4 des noyaux stables. Bien qu'il soit présent dans les sources du noyau, cette méthode de création dynamique des périphériques n'a jamais reçu un support inconditionnel des développeurs du noyau.

Le principal problème de l'approche adopté par `devfs` était la façon dont il gérait la détection, la création et le nommage des périphériques. Ce dernier problème, le nommage des périphériques, était peut-être le plus critique. Il est généralement entendu que s'il est possible de configurer les noms des périphériques, alors la politique de nommage des périphériques revient à l'administrateur du système, et non imposée par quelque développeur. Le système de fichiers `devfs` souffrait aussi de restrictions particulières inhérentes à sa conception et qui ne pouvaient être corrigées sans une revue importante du noyau. `devfs` a aussi été marqué comme obsolète pendant une longue période, à cause d'un manque de maintenance, et a finalement été supprimé du noyau en juin 2006.

Avec le développement de la branche instable 2.5 du noyau, sortie ensuite avec la série 2.6 des noyaux stables, un nouveau système de fichiers virtuel appelé `sysfs` est arrivé. Le rôle de `sysfs` est d'exporter une vue de la configuration matérielle du système pour les processus en espace utilisateur. Avec cette représentation visible en espace utilisateur, la possibilité de développer un remplacement en espace utilisateur de `devfs` est devenu beaucoup plus réaliste.

9.3.2. Implémentation d'Udev

9.3.2.1. Sysfs

Le système de 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 leurs objets avec le `sysfs` (en interne, devtmpfs) quand ils sont détectés par le noyau. Pour les pilotes compilés en tant que modules, cet enregistrement surviendra quand le module sera chargé. Une fois que le système de fichier `sysfs` est monté (sur `/sys`), les données enregistrées par les pilotes avec `sysfs` sont disponibles pour les processus en espace utilisateur ainsi que pour udevd pour traitement (y compris des modifications aux nœuds de périphériques).

9.3.2.2. Création de nœuds de périphérique

Les fichiers de périphérique sont créés par le noyau avec le système de fichiers `devtmpfs`. Tout pilote souhaitant enregistrer un nœud de périphérique ira dans le `devtmpfs` (par le cœur du pilote) pour le faire. Quand une instance `devtmpfs` est montée sur `/dev`, le nœud de périphérique sera créé dès le départ avec un nom, des droits et un propriétaire figés.

Peu de temps après, le noyau enverra un uevent à **udevd**. À partir des règles indiquées dans les fichiers contenus dans les répertoires `/etc/udev/rules.d`, `/usr/lib/udev/rules.d` et `/run/udev/rules.d`, **udevd** créera les liens symboliques supplémentaires vers le nœud de périphérique, ou bien il modifiera ses droits, son propriétaire ou son groupe, ou l'entrée dans la base de données interne d'**udevd** concernant cet objet.

Les règles de ces trois répertoires sont numérotées et les trois répertoires sont fusionnés. Si **udevd** ne peut pas trouver de règles pour le périphérique qu'il crée, il en donnera la propriété et les droits à n'importe quel `devtmpfs` utilisé au départ.

9.3.2.3. Chargement d'un module

Il se peut que les pilotes des périphériques compilés en module aient aussi des alias compilés. Les alias sont visibles dans la sortie du programme **modinfo** et sont souvent liés aux identifiants de bus spécifiques des périphériques pris en charge par un module. Par exemple, le pilote *snd-fm801* prend en charge les périphériques PCI ayant l'ID fabricant 0x1319 et l'ID de périphérique 0x0801 a aussi un alias `pci:v00001319d00000801sv*sd*bc04sc01i*`. Pour la plupart des périphériques, le pilote du bus définit l'alias du pilote qui gérerait le périphérique via `sysfs`. Par exemple, le fichier `/sys/bus/pci/devices/0000:00:0d.0/modalias` pourrait contenir la chaîne `pci:v00001319d00000801sv00001319sd00001319bc04sc01i00`. Les règles par défaut fournies par Udev feront que **udevd** appellera **/sbin/modprobe** avec le contenu de la variable d'environnement de l'uevent `MODALIAS` (qui devrait être la même que le contenu du fichier `modalias` dans sysfs), donc chargera tous les modules dont les alias correspondent à cette chaîne après les expansions génériques.

Dans cet exemple, cela signifie que, outre *snd-fm801*, le pilote obsolète (et non désiré) *forte* sera chargé s'il est disponible. Voir ci-dessous les moyens d'empêcher le chargement des modules indésirables.

Le noyau lui-même est aussi capable de charger des modules de protocole réseau, de prise en charge pour des systèmes de fichiers et de prise en charge native des langues sur demande.

9.3.2.4. Gestion des périphériques dynamiques ou montables à chaud

Lorsque vous connectez un périphérique, comme un lecteur MP3 USB, le noyau reconnaît que le périphérique est maintenant connecté et génère un uevent. Cet uevent est alors géré par **udevd** comme décrit ci-dessus.

9.3.3. Problèmes avec le chargement des modules et la création des périphériques

Il existe quelques problèmes connus pour la création automatique des nœuds de périphériques.

9.3.3.1. Un module noyau n'est pas chargé automatiquement

Udev ne chargera un module que s'il possède un alias spécifique au bus et que le pilote du bus envoie correctement les alias nécessaires vers `sysfs`. Autrement, il faut organiser le chargement des modules par d'autres moyens. Avec Linux-6.16.1, udev est connu pour charger les pilotes correctement écrits pour les périphériques INPUT, IDE, PCI, USB, SCSI, SERIO et FireWire.

Pour déterminer si le pilote du périphérique dont vous avez besoin prend en charge udev, lancez **modinfo** avec le nom du module en argument. Puis, essayez de localiser le répertoire du périphérique sous `/sys/bus` et vérifiez s'il y a un fichier `modalias`.

Si le fichier `modalias` existe dans `sysfs`, alors le pilote prend en charge le périphérique et peut lui parler directement, mais s'il n'a pas d'alias, c'est un bogue dans le pilote. Chargez le pilote sans l'aide d'udev et attendez que le problème soit corrigé ultérieurement.

S'il n'y a pas de fichier `modalias` dans le bon répertoire sous `/sys/bus`, cela signifie que les développeurs du noyau n'ont pas encore ajouté de prise en charge de modalias à ce type de bus. Avec Linux-6.16.1, c'est le cas pour les bus ISA. Attendez que ce problème soit corrigé dans les versions ultérieures du noyau.

Udev n'a pas du tout pour but de charger des pilotes « wrapper » (qui emballent un autre pilote) comme *snd-pcm-oss* et des pilotes non matériels comme *loop*.

9.3.3.2. Un module du noyau n'est pas chargé automatiquement et udev n'est pas prévu pour le charger

Si le module « enveloppe » n'améliore que la fonctionnalité fournie par un autre module (comme *snd-pcm-oss* améliore la fonctionnalité de *snd-pcm* en rendant les cartes son disponibles pour les applications OSS), configurez **modprobe** pour charger l'enveloppe après qu'udev a chargé le module enveloppé. Pour cela, ajoutez une ligne « softdep » dans tous les fichiers `/etc/modprobe.d/<filename>.conf`. Par exemple :

```
softdep snd-pcm post: snd-pcm-oss
```

Remarquez que la commande « softdep » autorise aussi les dépendances `pre:`, ou un mélange de `pre:` et de `post:`. Voir la page de manuel de *modprobe.d(5)* pour plus d'informations sur la syntaxe et les possibilités de « softdep ».

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

9.3.3.3. Udev charge un module indésirable

Ne compilez pas le module, ou mettez-le en liste noire dans un fichier `/etc/modprobe.d/blacklist.conf` comme réalisé avec le module *forte* dans l'exemple ci-dessous :

```
blacklist forte
```

Les modules en liste noire peuvent toujours être chargés manuellement avec la commande explicite **modprobe**.

9.3.3.4. Udev crée un périphérique incorrect, ou un mauvais lien symbolique

Cela se produit habituellement si une règle correspond à un périphérique de façon imprévue. Par exemple, une règle lacunaire peut correspondre à un disque SCSI (comme désiré) et au périphérique SCSI générique du même fabricant (de façon incorrecte). Trouvez la règle défectueuse et affinez-la, à l'aide de la commande **udevadm info**.

9.3.3.5. Une règle Udev fonctionne de manière non fiable

Cela peut être une autre manifestation du problème précédent. Si ce n'est pas le cas, et si votre règle utilise les attributs de `sysfs`, il se peut que ce soit un problème de timing du noyau, sur le point d'être corrigé dans les noyaux ultérieurs. Pour le moment, vous pouvez contourner cela en créant une règle qui attend l'attribut `sysfs` utilisé et en le mettant dans le fichier `/etc/udev/rules.d/10-wait_for_sysfs.rules` (créez ce fichier s'il n'existe pas). Merci d'informer la liste de développement de LFS si vous faites ainsi et que cela vous aide.

9.3.3.6. Udev ne crée pas de périphérique

Les textes suivants supposent que le pilote est compilé statiquement dans le noyau ou bien déjà chargé comme module et que vous avez vérifié que udev ne crée pas de périphérique mal nommé.

Udev n'a pas les informations pour créer un nœud si un pilote noyau n'exporte pas ses informations vers `sysfs`. C'est le plus souvent le cas des pilotes tiers ne provenant pas du noyau. Créez un nœud de périphérique statique dans `/usr/lib/udev/devices` avec les numéros majeurs/mineurs appropriés (regardez le fichier `devices.txt` dans la documentation du noyau du vendeur du pilote tiers). Le nœud statique sera copié dans `/dev` par **udev**.

9.3.3.7. L'ordre de nommage des périphériques change de manière aléatoire après le redémarrage

Cela est dû au fait qu'udev, par nature, gère les uevents et charge les modules en parallèle, donc dans un ordre imprévisible. Cela ne sera jamais « corrigé ». Vous ne devriez pas supposer que les noms des périphériques du noyau sont stables. Créez plutôt vos propres règles qui rendent les liens symboliques stables basés sur des attributs stables du périphérique, comme une série de nombres ou la sortie de divers utilitaires *_id installés par udev. Voir la Section 9.4, « Gérer les périphériques » et la Section 9.5, « Configuration générale du réseau » pour des exemples.

9.3.4. Lecture utile

Des documentations supplémentaires sont disponibles sur les sites suivants :

- A Userspace Implementation of `devfs` *http://www.kroah.com/linux/talks/ols_2003_udev_paper/Reprint-Kroah-Hartman-OLS2003.pdf* (NdT : Une implémentation en espace utilisateur de devfs)
- The `sysfs` Filesystem *https://www.kernel.org/pub/linux/kernel/people/mochel/doc/papers/ols-2005/mochel.pdf* (NdT : Le système de fichiers `sysfs`)

9.4. Gérer les périphériques

9.4.1. Périphériques réseaux

Udev, par défaut, nomme les périphériques réseaux à partir des données du logiciel embarqué/BIOS ou des caractéristiques physiques comme leur bus, leur slot ou leur adresse MAC. Le but de cette convention de nommage est de s'assurer que les périphériques réseaux sont nommés de façon cohérente et non en fonction du moment où la carte réseau a été trouvée. Sur des anciennes versions de Linux—sur des ordinateurs avec deux cartes réseaux Intel et Realtek, par exemple, il se peut que la carte réseau Intel soit nommée eth0 et la Realtek soit nommée eth1. Au redémarrage, les cartes sont parfois numérotées en sens inverse.

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

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

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

9.4.1.2. Créer des règles Udev personnalisées

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

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

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

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

Note

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

Le fichier commence par un bloc en commentaire suivi de deux lignes pour chaque carte réseau. La première ligne d'une carte réseau est une description commentée indiquant les ID matériel (comme l'ID PCI du fabricant et du périphérique s'il s'agit d'une carte PCI), et le pilote (entre parenthèses, si le pilote n'est pas détecté). L'ID matériel et le pilote ne déterminent pas le nom à donner à une interface ; ces informations ne sont présentes qu'à titre informatif. La deuxième ligne est la règle udev correspondant à la carte réseau. Cette deuxième règle affecte le nom à l'interface.

Toutes les règles udev se composent de mots-clefs séparées par des virgules et éventuellement des espaces. Voici les mots-clefs avec leurs explications :

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

La valeur de `NAME` est une partie très importante. Assurez-vous de connaître le nom affecté à chacune de vos cartes réseaux avant de continuer, et utilisez cette valeur `NAME` au moment de créer les fichiers de configuration.

Même si vous créez le fichier de règles personnalisé, udev peut toujours assigner un ou plusieurs noms alternatifs à la carte réseau en fonction de ses caractéristiques physiques. Si une règle udev personnalisée renomme certaines cartes réseau avec un nom déjà assigné en tant que nom alternatif d'une autre carte, la règle udev échouera. Si ce problème arrive, vous pouvez créer le fichier de configuration `/etc/udev/network/99-default.link` avec une autre politique d'assignation, qui remplace le fichier de configuration par défaut `/usr/lib/udev/network/99-default.link` :

```
sed -e '/^AlternativeNamesPolicy/s/=.*$/=/'  \
       /usr/lib/udev/network/99-default.link \
     > /etc/udev/network/99-default.link
```

9.4.2. Liens symboliques vers le CD-ROM

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

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

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

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

Important

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

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

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

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

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

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

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

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

9.4.3. Gérer des périphériques dupliqués

Comme expliqué à la Section 9.3, « Manipulation des périphériques et modules, » l'ordre dans lequel les périphériques ayant la même fonction apparaissent dans `/dev` est essentiellement aléatoire. Par exemple, si vous avez une webcam USB et un récepteur TV, parfois `/dev/video0` renvoie à la webcam, et `/dev/video1` renvoie au récepteur, et parfois après un redémarrage l'ordre d'apparition des périphériques s'inverse. Pour toutes les classes de matériel sauf les cartes son et les cartes réseau, ceci peut se corriger en créant des règles udev pour créer des liens symboliques constants. Le cas des cartes réseau est traité séparément dans Section 9.5, « Configuration générale du réseau, » et vous pouvez trouver la configuration des cartes son dans *BLFS*.

Pour chacun des périphériques susceptibles d'avoir ce problème (même si le problème n'apparaît pas dans votre distribution Linux actuelle), trouvez le répertoire correspondant sous `/sys/class` ou `/sys/block`. Pour les périphériques vidéo, cela peut être `/sys/class/video4linux/videoX`. Calculez les attributs qui identifient de façon unique un périphérique (normalement basé sur l'ID du fabricant et du produit ou les numéros de série) :

```
udevadm info -a -p /sys/class/video4linux/video0
```

Puis, écrivez des règles qui créent les liens symboliques, comme :

```
cat > /etc/udev/rules.d/83-duplicate_devs.rules << "EOF"

# Persistent symlinks for webcam and tuner
KERNEL=="video*", ATTRS{idProduct}=="1910", ATTRS{idVendor}=="0d81", SYMLINK+="webcam"
KERNEL=="video*", ATTRS{device}=="0x036f",  ATTRS{vendor}=="0x109e", SYMLINK+="tvtuner"

EOF
```

Il en résulte que les périphériques `/dev/video0` et `/dev/video1` renvoient encore de manière aléatoire au tuner et à la webcam (et donc ne devrait jamais être utilisé directement), mais certains des liens symboliques `/dev/tvtuner` et `/dev/webcam` pointent toujours vers le bon périphérique.

9.5. Configuration générale du réseau

9.5.1. Création des fichiers de configuration d'interface réseau

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

Note

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

Les noms d'interface dépendent de l'implémentation et de la configuration du démon udev exécuté sur le système. Le démon udev de LFS (installé au Section 8.76, « Udev de Systemd-257.8 ») ne sera pas exécuté à moins de démarrer le système LFS. Il n'est donc pas fiable de déterminer les noms d'interface utilisés dans le système LFS en exécutant ces commandes sur la distribution hôte, *même dans l'environnement chroot*.

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

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

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

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

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

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

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

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

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

9.5.2. Créer le fichier /etc/resolv.conf

Le système aura besoin d'un moyen pour obtenir un résolveur DNS pour résoudre les noms de domaines Internet en adresse IP, et vice-versa. Ceci se fait en plaçant les adresses IP des serveurs DNS, disponibles auprès du FAI ou de l'administrateur système, dans /etc/resolv.conf. Créez ce fichier en lançant :

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

domain <Votre nom de domaine>
nameserver <Adresse IP du DNS primaire>
nameserver <Adresse IP du DNS secondaire>

# Fin de /etc/resolv.conf
EOF
```

Le paramètre domain peut être omis ou remplacé par un paramètre search. Voir la page de manuel de resolv.conf pour plus de détails.

Remplacez *<Adresse IP du DNS>* par l'adresse IP du DNS le plus approprié pour la configuration. Il y aura souvent plus d'une entrée (les serveurs secondaires sont utiles en cas d'indisponibilité du premier). Si vous avez seulement besoin ou si vous voulez seulement un serveur DNS, supprimez la seconde ligne *nameserver* du fichier. L'adresse IP pourrait aussi être un routeur sur le réseau local.

Note

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

9.5.3. Configurer le nom d'hôte du système

Pendant le processus de démarrage, le fichier /etc/hostname est utilisé pour donner un nom d'hôte au système.

Créez le fichier /etc/network et saisissez le nom du système en lançant :

```
echo "<lfs>" > /etc/hostname
```

<lfs> doit être remplacé par le nom de l'ordinateur. Ne saisissez pas le FQDN ici. Cette information sera saisie dans le fichier `/etc/hosts`.

9.5.4. Personnaliser le fichier /etc/hosts

Choisissez un nom de domaine pleinement qualifié (*fully-qualified domain name*, ou FQDN) et les alias possibles à déclarer dans le fichier `/etc/hosts`. Si vous utilisez une adresse IP statique, vous devrez décider de celle-ci. La syntaxe du fichier hosts est :

```
IP_address myhost.example.org aliases
```

Sauf si votre ordinateur doit être visible à partir d'Internet (c-à-d que c'est un domaine enregistré et un bloc d'adresses IP valide—ce que la plupart des utilisateurs n'ont pas), assurez-vous que l'adresse IP se trouve dans la plage d'adresses réservée aux réseaux privés. Les plages valides sont :

```
Plage d'adresses réseau privés      Préfixe normal
     10.0.0.1 - 10.255.255.254                   8
    172.x.0.1 - 172.x.255.254                   16
  192.168.y.1 - 192.168.y.254                   24
```

x peut être un nombre compris entre 16 et 31. y peut être un nombre compris entre 0 et 255.

Une adresse IP privée valide pourrait être 192.168.1.2.

Si l'ordinateur doit être visible sur Internet, un FQDN valide peut être le nom de domaine lui-même ou une chaine composée d'un préfixe (souvent le nom d'hôte) et du nom de domaine séparés du charactère « . ». Ensuite, vous devez contacter votre fournisseur de nom de domaine pour pouvoir résoudre le FQDN vers votre adresse IP publique.

Même si l'ordinateur n'est pas visible sur Internet, un FQDN est toujours requis par certains programmes, comme les MTA, pour fonctionner correctement. Un FQDN spécial, `localhost.localdomain`, peut être utilisé pour cela.

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

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

127.0.0.1 localhost.localdomain localhost
127.0.1.1 <FQDN> <HOSTNAME>
<192.168.1.2> <FQDN> <HOSTNAME> [alias1] [alias2 ...]
::1       localhost ip6-localhost ip6-loopback
ff02::1   ip6-allnodes
ff02::2   ip6-allrouters

# End /etc/hosts
EOF
```

Les valeurs *<192.168.1.2>*, *<FQDN>* et *<HOSTNAME>* doivent être remplacées suivant les contraintes et les besoins spécifiques (si la machine se voit affecter une adresse IP par un administrateur réseau/système et que cette machine est connectée à un réseau existant). Vous pouvez omettre le ou les noms d'alias facultatifs.

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

9.6.1. Comment fonctionnent les scripts de démarrage de System V ?

Linux utilise un outil de démarrage spécial nommé SysVinit, qui se base sur une série de *niveaux d'exécution*. Le processus de démarrage peut varier d'un système à un autre, et donc si il fonctionne sur une distribution Linux en particulier, il n'est pas garanti qu'il fonctionnera de la même matière dans LFS. LFS a son propre fonctionnement, mais il respecte généralement les standards établis.

Il existe un autre processus de démarrage appelé **systemd**. Nous n'aborderons pas ce processus plus en détail. Pour plus d'informations, consultez *https://www.linux.com/training-tutorials/understanding-and-using-systemd/*.

SysVinit (qui appellerons « init » à partir de maintenant) fonctionne avec des niveaux d'exécution. Ils sont au nombre de sept, numérotés de 0 à 6. (En réalité, il en existe d'autres, mais ils sont réservés à des cas spéciaux et ne sont généralement pas utilisés. Voir *init(8)* pour plus de détails.) Chacun d'entre eux correspond à des actions que l'ordinateur doit effectuer lorsqu'il démarre ou s'éteint. Le niveau d'exécution par défaut est 3. Voici les descriptions des différents niveaux d'exécution implémentés dans LFS :

```
0 : éteint l'ordinateur
1 : mode mono-utilisateur
2 : réservé pour la personnalisation, mais autrement identique au 3
3 : mode multi-utilisateur avec réseau
4 : réservé pour la personnalisation, mais autrement identique au 3
5 : identique au 4, habituellement utilisé pour la connexion GUI (comme la commande gdm de GNOME ou la comman
6 : redémarre l'ordinateur
```

Note

Traditionnellement, le niveau d'exécution 2 ci-dessus était qualifié de « mode multi-utilisateur sans réseau », comme c'était le cas il y a plusieurs années, quand plusieurs utilisateurs pouvaient se connecter à un système via les ports série. Aujourd'hui, cela n'a plus de sens et le niveau est donc « réservé ».

9.6.2. Configuration de SysVinit

Lors de l'initialisation du noyau, **init** est le premier programme qui se lance (s'il n'est pas remplacé sur la ligne de commande). Ce programme lit le fichier d'initialisation `/etc/inittab`. Créez ce fichier avec :

```
cat > /etc/inittab << "EOF"
# Début de /etc/inittab

id:3:initdefault:

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

l0:0:wait:/etc/rc.d/init.d/rc 0
l1:S1:wait:/etc/rc.d/init.d/rc 1
l2:2:wait:/etc/rc.d/init.d/rc 2
l3:3:wait:/etc/rc.d/init.d/rc 3
l4:4:wait:/etc/rc.d/init.d/rc 4
l5:5:wait:/etc/rc.d/init.d/rc 5
l6:6:wait:/etc/rc.d/init.d/rc 6

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

su:S06:once:/sbin/sulogin
s1:1:respawn:/sbin/sulogin

1:2345:respawn:/sbin/agetty --noclear tty1 9600
2:2345:respawn:/sbin/agetty tty2 9600
3:2345:respawn:/sbin/agetty tty3 9600
4:2345:respawn:/sbin/agetty tty4 9600
5:2345:respawn:/sbin/agetty tty5 9600
6:2345:respawn:/sbin/agetty tty6 9600

# Fin de /etc/inittab
EOF
```

Vous trouverez une explication de ce fichier d'initialisation dans la page de manuel de *inittab*. Dans LFS, **rc** est la commande clé. Le fichier d'initialisation ci-dessus demande à **rc** de lancer tous les scripts commençant par un S dans le répertoire `/etc/rc.d/rcS.d`, puis tous les scripts commençant par un S du répertoire `/etc/rc.d/rc?.d`, où le point d'interrogation est spécifié par la valeur initdefault.

Par commodité, le script **rc** lit une bibliothèque de fonctions dans le répertoire `/lib/lsb/init-functions`. Cette bibliothèque lit aussi un fichier de configuration facultatif, `/etc/sysconfig/rc.site`. Tous les paramètres de configuration du système décrits dans les sections suivantes peuvent être placés dans ce fichier, permettant de rassembler tous les paramètres système dans un seul fichier.

Pour faciliter le débogage, le script de fonctions enregistre également toute la sortie dans `/run/var/bootlog`. Le répertoire `/run` étant un tmpfs, ce fichier n'est pas persistant entre les redémarrages. Cependant, il est ajouté au fichier plus permanent `/var/log/boot.log` à la fin du processus de démarrage.

9.6.2.1. Modifier les niveaux d'exécution

La commande **init *<niveau_exécution>*** permet de modifier le niveau d'exécution (*<niveau_exécution>* correspond au niveau d'exécution souhaité). Par exemple, pour redémarrer l'ordinateur, un utilisateur peut exécuter la commande **init 6** qui est un alias de la commande **reboot**. De même, **init 0** est un alias de la commande **halt**.

Il existe un certain nombre de répertoires sous `/etc/rc.d` qui ressemblent à `rc?.d` (le point d'interrogation correspond au le numéro du niveau d'exécution) et `rcS.d`, tous contenant un certain nombre de liens symboliques. Certains liens commencent par un *K*, les autres par un *S*, mais ils comportent tous deux chiffres après la lettre initiale. Le K signifie l'arrêt (kill) d'un service et le S son démarrage (start). Les chiffres déterminent l'ordre dans lequel les scripts sont exécutés, de 00 à 99 : plus ce nombre est petit, plus tôt le script correspondant est exécuté. Quand **init** bascule sur un autre niveau d'exécution, les services appropriés sont soit lancés soit arrêtés, suivant le niveau d'exécution choisi.

Les vrais scripts sont dans `/etc/rc.d/init.d`. Ce sont eux qui font le vrai travail et les liens symboliques pointent tous vers eux. Les liens K et les liens S pointent vers le même script dans `/etc/rc.d/init.d`. Ceci est dû au fait que les scripts peuvent être appelés avec différents paramètres comme *`start`*, *`stop`*, *`restart`*, *`reload`* et *`status`*. Quand un lien K est rencontré, le script approprié est lancé avec l'argument *`stop`*. Quand un lien S est rencontré, le script approprié est lancé avec l'argument *`start`*.

Voici les descriptions de ce que font les arguments des scripts :

`start`
: Le service est lancé.

`stop`
: Le service est arrêté.

`restart`
: Le service est arrêté puis relancé.

`reload`
: La configuration du service est mise à jour. Cette commande est utilisée après la modification du fichier de configuration d'un service, quand le service n'a pas besoin d'être redémarré.

`status`
: Indique si le service est en cours d'exécution ainsi que les PID associés.

Vous êtes libre de modifier la façon dont le processus de démarrage fonctionne : après tout, c'est votre système LFS. Les fichiers donnés le sont à titre d'exemples.

9.6.3. Les scripts de démarrage Udev

Le script de démarrage `/etc/rc.d/init.d/udev` lance **udevd**, déclenche les périphériques « coldplug » déjà créés par le noyau et attend que les règles se terminent. Le script désactive aussi le gestionnaire d'uevent dans les réglages par défaut du fichier `/sbin/hotplug` . Cette action est due au fait que le noyau n'a plus besoin de faire appel à un binaire externe. Par contre, **udevd** va écouter sur un socket netlink les uevents récupérés par le noyau.

Le script **/etc/rc.d/init.d/udev_retry** se charge de re-déclencher les événements des sous-systèmes dont les règles s'appuient sur des systèmes de fichiers non montés jusqu'à ce que le script **mountfs**soit lancé (en particulier, `/usr` et `/var` peuvent avoir cet effet). Ce script s'exécute après le script **mountfs**, donc ces règles (si elles sont re-déclenchées)

doivent s'appliquer la deuxième fois. Le script se configure à partir du fichier `/etc/sysconfig/udev_retry` ; tout mot qui n'est pas un commentaire dans ce fichier est considéré comme un nom de sous-système à déclencher lors d'un nouvel essai. Pour trouver le sous-système d'un périphérique, utilisez la commande **udevadm info --attribute-walk <périphérique>** (<périphérique> étant un chemin absolu dans /dev ou /sys, comme /dev/sr0 ou /sys/class/rtc).

Pour plus d'informations sur le chargement des modules du noyau et udev, consultez Section 9.3.2.3, « Chargement d'un module. »

9.6.4. Configurer l'horloge système

Le script **setclock** lit l'heure à partir de l'horloge matérielle, aussi appelée horloge BIOS ou *Complementary Metal Oxide Semiconductor* (CMOS). Si l'horloge matérielle est réglée sur UTC, ce script convertira l'heure de l'horloge matérielle en heure locale à l'aide du fichier `/etc/localtime` (qui indique au programme **hwclock** le fuseau horaire de l'utilisateur). Il n'y a aucun moyen de détecter si l'horloge matérielle est réglée sur UTC, donc vous devez configurer ce paramètre manuellement.

Le script **setclock** se lance via udev quand le noyau détecte le matériel au démarrage. Vous pouvez aussi le lancer manuellement avec le paramètre stop pour stocker l'heure du système dans l'horloge CMOS.

Si vous ne vous rappelez pas si l'horloge matérielle est réglée sur UTC, exécutez la commande **`hwclock --localtime --show`**. Elle affichera l'heure actuelle en se basant sur l'horloge matérielle. Si elle correspond à ce qu'indique votre montre, l'horloge matérielle est réglée à l'heure locale. Si la sortie de **hwclock** ne correspond pas à l'heure locale, il y a des chances pour qu'il s'agisse de l'heure UTC. Pour le vérifier, ajoutez ou retirez le bon nombre d'heures du fuseau horaire de l'heure affichée par **hwclock**. Par exemple, si vous êtes dans le fuseau MST, aussi appelé GMT -0700, ajoutez sept heures à l'heure locale.

Changez la valeur de la variable `UTC` ci-dessous en *`0`* (zéro) si l'horloge matérielle *n'est pas* réglée sur l'heure UTC.

Créez un nouveau fichier `/etc/sysconfig/clock` en lançant ce qui suit :

```
cat > /etc/sysconfig/clock << "EOF"
# Début de /etc/sysconfig/clock

UTC=1

# Mettez ici les options que vous pourriez avoir besoin de donner à hwclock,
# comme le type de l'horloge matérielle de la machine pour les Alphas.
CLOCKPARAMS=

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

Une bonne astuce qui explique la gestion de l'heure sur LFS est disponible sur *http://www.fr.linuxfromscratch.org/view/astuces/heure.txt*. Elle traite de sujets tels que les fuseaux horaires, UTC et la variable d'environnement `TZ`.

Note

Vous pouvez aussi régler les paramètres CLOCKPARAMS et UTC dans le fichier `/etc/sysconfig/rc.site`.

9.6.5. Configurer la Console Linux

Cette section traite de la configuration du script de démarrage **console**, qui initialise la configuration du clavier, la police de la console et le niveau du journal du noyau. Si vous n'utilisez pas les caractères non-ASCII (par exemple le symbole du copyright, de la livre sterling et de l'euro) et que votre clavier est nord-américain, vous pouvez passer une grande partie de cette section. Sans le fichier de configuration (ou des options équivalentes dans le fichier `rc.site`), le script de démarrage **console** ne fera rien.

Le script **console** lit les informations de configuration du fichier `/etc/sysconfig/console`. Il décide de la configuration du clavier et de la police de la console à utiliser. Différents guides pratiques spécifiques à votre langue peuvent vous être d'une grande aide (voir *https://tldp.org/HOWTO/HOWTO-INDEX/other-lang.html*). Si vous avez toujours des doutes, jetez un œil au contenu des répertoires `/usr/share/keymaps` et `/usr/share/consolefonts` pour consulter des configurations de clavier et des polices d'écran valides. Lisez les pages de manuel *loadkeys(1)* et *setfont(8)* pour déterminer les arguments à passer à ces programmes.

Le fichier `/etc/sysconfig/console` devrait contenir des lignes de la forme : `VARIABLE=valeur`. Les variables suivantes sont reconnues :

LOGLEVEL

Cette variable spécifie le niveau de traçage pour les messages du noyau envoyés à la console, selon le paramétrage de **dmesg -n**. Les niveaux valides vont de `1` (aucun message) à `8`. Le niveau par défaut est `7`.

KEYMAP

Cette variable spécifie les arguments du programme **loadkeys**, généralement, le nom de la configuration du clavier à charger, comme `it`. Si cette variable n'est pas initialisée, le script de démarrage ne lancera pas le programme **loadkeys** et la configuration du clavier par défaut du noyau sera utilisée.Peu de configurations du clavier ont plusieurs versions avec le même nom (cz et ses variantes en qwerty/ et en qwertz/, es en olpc/ et en qwerty/, et trf en fgGlod/ et en qwerty/). Dans ce cas, le répertoire parent doit être spécifié (par exemple qwerty/es) pour s'assurer que la configuration de clavier adaptée est chargée.

KEYMAP_CORRECTIONS

Cette variable (rarement utilisée) spécifie les arguments du second appel au programme **loadkeys**. Elle est utile si la configuration du clavier par défaut n'est pas totalement satisfaisante et que vous devez faire un petit ajustement. Par exemple, pour inclure le symbole euro dans une configuration de clavier qui ne le prend pas en charge, réglez cette variable à `euro2`.

FONT

Cette variable spécifie les arguments du programme **setfont**. Généralement, elle inclut le nom de la police, `-m` et le nom de la configuration du clavier à charger. Par exemple, pour charger la police « lat1-16 » avec la configuration du clavier « 8859-1 » (comme aux États-Unis), réglez cette variable à `lat1-16 -m 8859-1`. En mode UTF-8, le noyau utilise la configuration du clavier pour convertir des codes de touches 8 bits en UTF-8. Ainsi, vous devriez initialiser l'argument du paramètre "-m" à l'encodage des codes de touches composés dans la configuration de clavier.

UNICODE

Réglez cette variable à `1`, `yes` ou `true` afin de mettre la console en mode UTF-8. Cette variable est utile pour les paramètres géographiques basés sur UTF-8, mais elle peut être dangereuse dans le cas contraire.

LEGACY_CHARSET

Pour beaucoup de types de clavier, il n'existe pas de configuration de clavier pour le stock Unicode dans le paquet Kbd. Le script de démarrage **console** convertira une configuration de clavier disponible en UTF-8 au vol si cette variable est réglée à l'encodage de la configuration du clavier non UTF-8 disponible.

Quelques exemples :

- Nous allons utiliser `C.UTF-8` comme paramètre linguistique pour les sessions interactives dans la console Linux dans Section 9.7, « Configuration des paramètres régionaux du système, », donc nous devrions paramétrer `UNICODE` à `1`. De plus, les polices de console fournies par le paquet Kbd contenant les glyphes pour tous les caractères des messages des programmes dans le paramètre linguistique `C.UTF-8` sont `LatArCyrHeb*.psfu.gz`,

`LatGrkCyr*.psfu.gz`, `Lat2-Terminus16.psfu.gz` et `pancyrillic.f16.psfu.gz` dans `/usr/share/consolefonts` (les autres polices fournies n'ont pas certains caractères comme le guillemet Unicode ou le tiret quadratin). Pour utiliser l'une d'entre elles, par exemple `Lat2-Terminus16.psfu.gz` comme police de console par défaut :

```
cat > /etc/sysconfig/console << "EOF"
# Début de /etc/sysconfig/console

UNICODE="1"
FONT="Lat2-Terminus16"

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

- Pour une initialisation non Unicode, en général seules les variables KEYMAP et FONT sont nécessaires. Par exemple, pour l'initialisation en polonais, on utiliserait :

```
cat > /etc/sysconfig/console << "EOF"
# Début de /etc/sysconfig/console

KEYMAP="pl2"
FONT="lat2a-16 -m 8859-2"

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

- Comme mentionné ci-dessus, il est parfois nécessaire d'ajuster légèrement la configuration de clavier stockée. L'exemple suivant ajoute le symbole euro à la configuration allemande du clavier :

```
cat > /etc/sysconfig/console << "EOF"
# Begin /etc/sysconfig/console

KEYMAP="de-latin1"
KEYMAP_CORRECTIONS="euro2"
FONT="lat0-16 -m 8859-15"
UNICODE="1"

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

- Ce qui suit est un exemple où l'Unicode est activé pour le bulgare, et où une configuration du clavier UTF-8 par défaut existe :

```
cat > /etc/sysconfig/console << "EOF"
# Début de /etc/sysconfig/console

UNICODE="1"
KEYMAP="bg_bds-utf8"
FONT="LatArCyrHeb-16"

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

- Du fait de l'utilisation d'une police 512 glyphes LatArCyrHeb-16 dans l'exemple précédent, les couleurs claires ne sont plus disponibles sur la console Linux à moins d'utiliser un framebuffer. Si vous voulez avoir les couleurs claires sans framebuffer et que vous n'avez pas forcément besoin des caractères n'appartenant pas à votre langue, il est encore possible d'utiliser une police 256 glyphes spécifique à votre langue, comme illustré ci-dessous :

```
cat > /etc/sysconfig/console << "EOF"
# Début de /etc/sysconfig/console

UNICODE="1"
KEYMAP="bg_bds-utf8"
FONT="cyr-sun16"

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

- L'exemple suivant illustre l'auto-conversion de la configuration de clavier d'ISO-8859-15 vers UTF-8 et l'activation des touches mortes en mode Unicode :

```
cat > /etc/sysconfig/console << "EOF"
# Début de /etc/sysconfig/console

UNICODE="1"
KEYMAP="de-latin1"
KEYMAP_CORRECTIONS="euro2"
LEGACY_CHARSET="iso-8859-15"
FONT="LatArCyrHeb-16 -m 8859-15"

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

- Certaines configurations du clavier comportent des touches mortes (c'est-à-dire des touches qui ne génèrent pas un caractère en elles-mêmes, mais qui permettent d'ajouter un accent au caractère généré par la touche suivante) ou définissent des règles de comportement (comme : « Appuyez sur Ctrl+. A E pour obtenir Æ » dans la configuration du clavier par défaut). Linux-6.16.1 n'interprète correctement les touches mortes et les règles de composition que lorsque les caractères sources qui seront composés ensemble sont multi-octets. Ce défaut n'affecte pas les configurations de clavier pour les langues européennes, car les accents sont ajoutés à des caractères ASCII non accentués, ou deux caractères ASCII sont composés ensemble. Cependant, en mode UTF-8, c'est un problème, comme pour le grec, où on a parfois besoin de placer un accent sur la lettre α. La solution consiste soit à éviter d'utiliser UTF-8, soit à installer le système de fenêtrage X qui n'a pas cette limitation dans sa gestion de l'entrée.
- Pour le chinois, le japonais, le coréen et certaines autres langues, la console Linux ne peut pas être configurée pour afficher les caractères nécessaires. Les utilisateurs qui ont besoin de ces langues doivent installer le système de fenêtrage X, dont les polices couvrent la plage de caractères nécessaire et qui a la méthode d'entrée adaptée (par exemple, SCIM prend en charge une grande variété de langues).

Note

Le fichier `/etc/sysconfig/console` ne contrôle que la localisation de la console texte de Linux. Cela n'a rien à voir avec le bon paramétrage du type de clavier et des polices du terminal dans le système de fenêtrage X, avec les sessions ssh ou une console série. Dans de telles situations, les limites mentionnées dans les deux derniers points de la liste ci-dessus ne s'appliquent pas.

9.6.6. Créer des fichiers au démarrage

Parfois, on veut créer des fichiers lors du démarrage. Par exemple, le répertoire `/tmp/.ICE-unix` est souvent requis. Vous pouvez le faire en créant une entrée dans le script de configuration `/etc/sysconfig/createfiles`. Le format de ce fichier est indiqué dans les commentaires du fichier de configuration par défaut.

9.6.7. Configuration du script sysklogd

Le script `sysklogd` invoque le programme **syslogd** faisant partie de l'initialisation par System V. L'option *`-m 0`* désactive la marque périodique que **syslogd** écrit par défaut dans les fichiers journaux toutes les 20 minutes. Si vous voulez activer cet horodatage, éditez `/etc/sysconfig/rc.site` et définissez la variable SYSKLOGD_PARMS à la valeur désirée. Par exemple, pour supprimer tous les paramètres, réglez la variable à la valeur null :

```
SYSKLOGD_PARMS=
```

Voir **`man syslogd`** pour plus d'options.

9.6.8. Le fichier rc.site

Le fichier facultatif `/etc/sysconfig/rc.site` contient les paramètres réglés automatiquement pour chaque script de démarrage de System V. Il peut également configurer les valeurs des fichiers `hostname`, `console` et `clock` du répertoire `/etc/sysconfig/`. Si les variables associées se trouvent à la fois dans ces fichiers distincts et dans le fichier `rc.site`, les valeurs contenues dans les fichiers spécifiques au script sont prioritaires.

`rc.site` contient aussi des paramètres pour personnaliser d'autres aspects du processus de démarrage. Le réglage de la variable IPROMPT permettra un lancement sélectif des scripts de démarrage. D'autres options sont décrites dans les commentaires du fichier. La version par défaut du fichier est ci-dessous :

```
# rc.site
# Optional parameters for boot scripts.

# Distro Information
# These values, if specified here, override the defaults
#DISTRO="Linux From Scratch" # The distro name
#DISTRO_CONTACT="lfs-dev@lists.linuxfromscratch.org" # Bug report address
#DISTRO_MINI="LFS" # Short name used in filenames for distro config

# Define custom colors used in messages printed to the screen

# Please consult `man console_codes` for more information
# under the "ECMA-48 Set Graphics Rendition" section
#
# Warning: when switching from a 8bit to a 9bit font,
# the linux console will reinterpret the bold (1;) to
# the top 256 glyphs of the 9bit font.  This does
# not affect framebuffer consoles

# These values, if specified here, override the defaults
#BRACKET="\\033[1;34m" # Blue
#FAILURE="\\033[1;31m" # Red
#INFO="\\033[1;36m"    # Cyan
#NORMAL="\\033[0;39m"  # Grey
#SUCCESS="\\033[1;32m" # Green
#WARNING="\\033[1;33m" # Yellow

# Use a colored prefix
# These values, if specified here, override the defaults
#BMPREFIX="     "
#SUCCESS_PREFIX="${SUCCESS}  *  ${NORMAL} "
#FAILURE_PREFIX="${FAILURE}*****${NORMAL} "
#WARNING_PREFIX="${WARNING} *** ${NORMAL} "

# Manually set the right edge of message output (characters)
# Useful when resetting console font during boot to override
# automatic screen width detection
#COLUMNS=120

# Interactive startup
```

```
#IPROMPT="yes" # Whether to display the interactive boot prompt
#itime="3"    # The amount of time (in seconds) to display the prompt

# The total length of the distro welcome string, without escape codes
#wlen=$(echo "Welcome to ${DISTRO}" | wc -c )
#welcome_message="Welcome to ${INFO}${DISTRO}${NORMAL}"

# The total length of the interactive string, without escape codes
#ilen=$(echo "Press 'I' to enter interactive startup" | wc -c )
#i_message="Press '${FAILURE}I${NORMAL}' to enter interactive startup"

# Set scripts to skip the file system check on reboot
#FASTBOOT=yes

# Skip reading from the console
#HEADLESS=yes

# Write out fsck progress if yes
#VERBOSE_FSCK=no

# Speed up boot without waiting for settle in udev
#OMIT_UDEV_SETTLE=y

# Speed up boot without waiting for settle in udev_retry
#OMIT_UDEV_RETRY_SETTLE=yes

# Skip cleaning /tmp if yes
#SKIPTMPCLEAN=no

# For setclock
#UTC=1
#CLOCKPARAMS=

# For consolelog (Note that the default, 7=debug, is noisy)
#LOGLEVEL=7

# For network
#HOSTNAME=mylfs

# Delay between TERM and KILL signals at shutdown
#KILLDELAY=3

# Optional sysklogd parameters
#SYSKLOGD_PARMS="-m 0"

# Console parameters
#UNICODE=1
#KEYMAP="de-latin1"
#KEYMAP_CORRECTIONS="euro2"
#FONT="lat0-16 -m 8859-15"
#LEGACY_CHARSET=
```

9.6.8.1. Personnaliser les scripts de démarrage et d'extinction

Les scripts de démarrage LFS démarrent et arrêtent un système de manière très efficace, mais vous pouvez faire quelques modifications dans le fichier rc.site pour améliorer davantage leur rapidité et ajuster les messages selon vos préférences. Pour ce faire, modifiez les paramètres du fichier `/etc/sysconfig/rc.site` ci-dessus.

- Pendant le script de démarrage `udev`, un appel à **udev settle** demande un certain temps pour s'achever. Ce délai peut être nécessaire, ou non, selon les périphériques présents dans le système. Si vous n'avez que des partitions simples et une seule carte Ethernet, le processus de démarrage n'aura probablement pas besoin d'attendre cette commande. Pour la passer, définissez la variable OMIT_UDEV_SETTLE=y.

- Le script de démarrage `udev_retry` lance aussi **udev settle** par défaut. Cette commande n'est nécessaire que si le répertoire `/var` est monté séparément. En effet, la vérification a besoin du fichier `/var/lib/hwclock/adjtime`. D'autres personnalisations peuvent nécessiter d'attendre qu'udev se termine mais dans beaucoup d'installations, ce n'est pas nécessaire. Passez la commande en définissant la variable OMIT_UDEV_RETRY_SETTLE=y.
- Par défaut, les vérifications des systèmes de fichiers sont sans message. Cela peut être vu comme un retard pendant le processus de démarrage. Pour activer la sortie de **fsck**, définissez la variable VERBOSE_FSCK=y.
- Lors du redémarrage, il se peut que vous vouliez passer complètement la vérification du système de fichiers, **fsck**. Vous pouvez pour cela soit créer le fichier `/fastboot`, soit redémarrer le système avec la commande **/sbin/shutdown -f -r now**. Inversement, vous pouvez forcer la vérification de tous les systèmes de fichiers en créant `/forcefsck` ou en exécutant **shutdown** avec le paramètre *`-F`* plutôt que *`-f`*.

  La définition de la variable FASTBOOT=y désactivera **fsck** lors du processus de démarrage jusqu'à ce qu'il soit supprimé. Ce n'est pas recommandé de façon permanente.
- En principe, tous les fichiers du répertoire `/tmp` sont effacés au moment du démarrage. Selon le nombre de fichiers ou de répertoires présents, cela peut provoquer un retard important dans le processus de démarrage. Pour passer la suppression de ces fichiers, définissez la variable SKIPTMPCLEAN=y.
- Lors de l'arrêt, le programme **init** envoie un signal TERM à chaque programme qu'il a démarré (comme agetty), attend une durée déterminée (par défaut, 3 secondes), puis envoie à chaque processus un signal KILL puis attend de nouveau. Ce processus se répète dans le script **sendsignals** pour tous les processus non terminés par leurs propres scripts. Le délai de **init** peut être défini en passant un paramètre. Par exemple, pour supprimer le délai de **init**, passez le paramètre -t0 lors de l'arrêt ou du redémarrage (comme **/sbin/shutdown -t0 -r now**). Le délai du script **sendsignals** peut être passé en définissant le paramètre KILLDELAY=0.

9.7. Configuration des paramètres régionaux du système

Certaines variables d'environnement sont nécessaires pour la prise en charge des langues naturelles. Les configurer a pour résultat :

- La sortie des programmes est traduite dans votre langue maternelle
- Le classement correct des caractères en lettres, chiffres et autres classes. Cela est nécessaire pour **bash** pour accepter correctement les caractères non-ASCII pour les paramètres linguistiques non anglais
- Le tri alphabétique correct pour les autres pays
- La taille de papier par défaut appropriée
- Le formatage correct de la monnaie, du temps et de la date

Remplacez *`<ll>`* ci-dessous par les deux lettres de la langue désirée (par exemple `fr`) et remplacez *`<CC>`* par les deux lettres du pays approprié (par exemple `FR`). *`<charmap>`* doit être remplacé par la table de caractères classique pour le paramètre linguistique choisi. On peut aussi ajouter des modificateurs facultatifs comme `@euro`.

La liste des paramètre régionaux pris en charge par Glibc peut être obtenue en exécutant la commande suivante :

```
locale -a
```

Les jeux de caractères peuvent aussi avoir certains alias, par exemple `ISO-8859-1` est aussi référencé comme `iso8859-1` et `iso88591`. Certaines applications ne peuvent pas gérer les différents synonymes (il est par exemple requis que `UTF-8` soit écrit `UTF-8` et pas `utf8`), il est donc plus sur de choisir le nom classique pour un paramètre linguistique particulier. Pour déterminer le nom classique, lancez la commande suivante, où *`<nom du paramètre linguistique>`* est la sortie donnée par **locale -a** pour votre langue préférée (`fr_FR.iso88591` dans notre exemple).

```
LC_ALL=<nom du paramètre linguistique> locale charmap
```

Pour le paramètre `fr_FR.iso885915`, la commande ci-dessus affichera :

```
ISO-8859-15
```

Cela résulte en un paramètre de régionalisation final de `fr_FR.ISO-8859-15`. Il est important que le paramètre trouvé en utilisant la syntaxe précédente soit testé avant d'être ajouté aux fichiers de démarrage de bash :

```
LC_ALL=<nom du paramètre linguistique> locale language
LC_ALL=<nom du paramètre linguistique> locale charmap
LC_ALL=<nom du paramètre linguistique> locale int_curr_symbol
LC_ALL=<nom du paramètre linguistique> locale int_prefix
```

Les commandes ci-dessus devraient afficher la langue, l'encodage des caractères utilisé par le paramètre, la monnaie locale ainsi que le préfixe de téléphone à composer afin d'entrer dans le pays. Si une de ces commandes précédentes échoue, cela signifie que votre paramètre n'a pas été installé dans le chapitre 8 ou n'est pas prise en charge par l'installation par défaut de Glibc.

```
locale: Cannot set LC_* to default locale: No such file or directory
```

Si cela arrive, vous devriez soit installer le paramètre linguistique désiré en utilisant la commande **localedef**, soit utiliser un autre paramètre. Les instructions suivantes supposent que vous n'avez pas reçu ces messages d'erreur de Glibc.

D'autres programmes ne vont aussi pas fonctionner correctement (mais ne vont pas nécessairement afficher un message d'erreur) si le nom du paramètre régional ne rencontre pas leurs attentes. Dans d'autres cas, rechercher comment d'autres distributions Linux prennent en charge votre paramètre régional peut fournir des informations utiles.

Le programme shell **/bin/bash** (à partir de maintenant appelé « le shell ») utilise un ensemble de fichiers de démarrage pour créer l'environnement dans lequel il s'exécute. Chaque fichier a un but précis et peut affecter différemment les environnements de connexion et interactifs. Les fichiers dans le répertoire `/etc` fournissent des paramètres génériques. Si un fichier équivalent existe dans le répertoire personnel, il peut remplacer les paramètres génériques.

Un shell de connexion interactif est démarré après une connexion réussie, avec **/bin/login**, en lisant le fichier `/etc/passwd`. Un shell interactif sans connexion est démarré par la ligne de commande (p. ex. `[invite]$`**/bin/bash**). Un shell non-interactif est en général présent quand un script shell est lancé. Il n'est pas interactif car il traite un script et n'attend pas d'entrées utilisateur entre les commandes.

Créez le fichier `/etc/profile` une fois les paramètres linguistiques déterminés pour indiquer le paramètre souhaité, mais indiquez le paramètre `C.UTF-8` à la place si le shell tourne sous la console Linux (pour éviter que les programmes n'affichent des caractères que la console Linux ne peut pas afficher) :

```
cat > /etc/profile << "EOF"
# Début de /etc/profile

for i in $(locale); do
  unset ${i%=*}
done

if [[ "$TERM" = linux ]]; then
  export LANG=C.UTF-8
else
  export LANG=<ll>_<CC>.<charmap><@modifiers>
fi

# Fin de /etc/profile
EOF
```

Les paramètres `C` (par défaut) et `en_US` (recommandé pour les utilisateurs anglophones des États-Unis) sont différents. `C` utilise le jeu de caractères US-ASCII 7-bit et traite les octets avec le bit de point fort à 1 comme des caractères invalides. C'est pourquoi, par exemple, la commande **ls** les remplace par des points d'interrogation sous ce paramètre. De plus, les messages contenant ces caractères envoyés avec Mutt ou Pine ne sont pas conformes aux RFC (le jeu de caractères dans le mail sortant est indiqué comme `unknown 8-bit`). Vous ne pouvez donc utiliser le paramètre régional `C` que si vous êtes sûr que vous n'aurez jamais besoins de caractère 8 bits.

9.8. Créer le fichier /etc/inputrc

Le fichier `inputrc` est un fichier de configuration pour la bibliothèque readline, qui fournit des fonctions d'édition lors de la saisie de commandes dans le terminal. Elle fonctionne en traduisant l'entrée du clavier en des actions spécifiques. Readline est utilisé par bash et par la plupart des autres shells ainsi que par de nombreuses autres applications.

La plupart des personnes n'ont pas besoin de fonctionnalités personnalisées, donc la commande ci-dessous crée un fichier `/etc/inputrc` global utilisé par tous ceux qui se connectent. Si vous décidez plus tard que vous avez besoin de surcharger les valeurs par défaut utilisateur par utilisateur, vous pouvez créer un fichier `.inputrc` dans le répertoire personnel de l'utilisateur avec les correspondances modifiées.

Pour plus d'informations sur l'édition du fichier `inputrc`, voir **info bash** à la section *Fichier d'initialisation Readline* (ou *Readline Init File*). **info readline** est aussi une bonne source d'informations.

Ci-dessous se trouve un fichier `inputrc` générique avec des commentaires expliquant l'utilité des différentes options. Remarquez que les commentaires ne peuvent pas être sur la même ligne que les commandes. Créez le fichier en utilisant la commande suivante :

```
cat > /etc/inputrc << "EOF"
# Début de /etc/inputrc
# Modifié par Chris Lynn <roryo@roryo.dynup.net>

# Permet à l'invite de commande de continuer sur la ligne suivante
set horizontal-scroll-mode Off

# Active l'entrée 8-bits
set meta-flag On
set input-meta On

# Désactive la suppression du 8ème bit
set convert-meta Off

# Garder le 8ème bit pour l'affichage
set output-meta On

# « none », « visible » ou « audible »
set bell-style none

# Toutes les indications qui suivent font correspondre la séquence
# d'échappement contenue dans le 1er argument à la fonction
"\eOd": backward-word
"\eOc": forward-word

# pour la console linux
"\e[1~": beginning-of-line
"\e[4~": end-of-line
"\e[5~": beginning-of-history
"\e[6~": end-of-history
"\e[3~": delete-char
"\e[2~": quoted-insert

# pour xterm
"\eOH": beginning-of-line
"\eOF": end-of-line

# pour Konsole
"\e[H": beginning-of-line
"\e[F": end-of-line

# Fin de /etc/inputrc
EOF
```

9.9. Créaction du fichier /etc/shells

Le fichier `shells` contient une liste des shells de connexion présents sur le système. Les applications utilisent ce fichier pour déterminer si un shell est valide. Pour chaque shell, une seule ligne doit être présente, contenant l'emplacement du shell relativement à la racine (/).

Par exemple, ce fichier est consulté par **chsh** pour déterminer si un utilisateur non privilégié peut modifier le shell de connexion de son compte. Si le nom de la commande n'est pas listé, l'utilisateur n'aura pas le droit d'en changer.

C'est nécessaire pour des applications telles que GDM qui ne peuplent pas le navigateur d'interface s'il ne peut pas trouver `/etc/shells`, ou les démons FTP qui interdisent traditionnellement aux utilisateurs l'accès avec des shells qui ne sont pas inclus dans ce fichier.

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

/bin/sh
/bin/bash

# Fin de /etc/shells
EOF
```

Chapitre 10. Rendre le système LFS amorçable

10.1. Introduction

Il est temps de rendre amorçable le système LFS. Ce chapitre traite de la création d'un fichier `fstab`, de la construction d'un noyau pour le nouveau système LFS et de l'installation du chargeur de démarrage GRUB afin que le système LFS puisse être sélectionné au démarrage.

10.2. Créer le fichier /etc/fstab

Le fichier `/etc/fstab` est utilisé par quelques programmes pour déterminer les systèmes de fichiers à monter par défaut, dans quel ordre, et lesquels doivent être vérifiés (recherche d'erreurs d'intégrité) avant le montage. Créez une nouvelle table des systèmes de fichiers comme ceci :

```
cat > /etc/fstab << "EOF"
# Début de /etc/fstab

# file system  mount-point    type     options             dump  fsck
#                                                                order

/dev/<xxx>     /              <fff>    defaults            1     1
/dev/<yyy>     swap           swap     pri=1               0     0
proc           /proc          proc     nosuid,noexec,nodev 0     0
sysfs          /sys           sysfs    nosuid,noexec,nodev 0     0
devpts         /dev/pts       devpts   gid=5,mode=620      0     0
tmpfs          /run           tmpfs    defaults            0     0
devtmpfs       /dev           devtmpfs mode=0755,nosuid    0     0
tmpfs          /dev/shm       tmpfs    nosuid,nodev        0     0
cgroup2        /sys/fs/cgroup cgroup2  nosuid,noexec,nodev 0     0

# Fin de /etc/fstab
EOF
```

Remplacez *<xxx>*, *<yyy>* et *<fff>* par les valeurs appropriées pour votre système, par exemple `sda2`, `sda5` et `ext4`. Pour tous les détails sur les six champs de cette table, consultez *fstab(5)*.

Les systèmes de fichiers ayant pour origine MS-DOS ou Windows (c-à-d. vfat, ntfs, smbfs, cifs, iso9660, udf) ont besoin d'une option spéciale, utf8, pour que les caractères non-ascii dans les noms de fichiers soient interprétés correctement. Pour les paramètres linguistiques non-utf-8, la valeur de `iocharset` devrait être la même que le jeu de caractères de votre locale, ajustée de telle sorte que le noyau la comprenne. Cela fonctionne si la définition du codage adéquat (que vous trouverez sous File systems -> Native Language Support lors de la configuration du noyau) a été compilée en dur dans le noyau ou en module. Cependant, si le jeu de caractères des paramètres linguistiques est UTF-8, l'option correspondante `iocharset=utf8` rendrait le système de fichier sensible à la casse. Pour corriger cela, utilisez l'option spéciale `utf8` au lieu de `iocharset=utf8` pour les paramètres linguistiques UTF-8. L'option « codepage » est aussi nécessaire aux systèmes de fichiers vfat et smbfs. Elle devrait être paramétrée pour correspondre à la page de code utilisée sous MS-DOS dans votre pays. Par exemple, pour monter des lecteurs flash USB, un utilisateur ru_RU.KOI8-R aurait besoin de ce qui suit dans la partie des options de sa ligne de montage dans `/etc/fstab` :

```
noauto,user,quiet,showexec,codepage=866,iocharset=koi8r
```

Le fragment d'options correspondantes pour les utilisateurs ru_RU.UTF-8 est :

```
noauto,user,quiet,showexec,codepage=866,utf8
```

Remarquez que l'utilisation de `iocharset` se fait par défaut pour `iso8859-1` (ce qui laisse le système de fichiers insensible à la casse) et l'option `utf8` dit au noyau de convertir les noms de fichiers en UTF-8 pour qu'ils puissent être interprétés dans les paramètres linguistiques UTF-8.

Il est aussi possible de spécifier les valeurs de page de code et de codage entrée/sortie (iocharset) par défaut pour certains systèmes de fichiers pendant la configuration du noyau. Les paramètres pertinents sont nommés « Default NLS Option » (`CONFIG_NLS_DEFAULT`), « Default Remote NLS Option » (`CONFIG_SMB_NLS_DEFAULT`), « Default codepage for FAT » (`CONFIG_FAT_DEFAULT_CODEPAGE`) et « Default iocharset for FAT » (`CONFIG_FAT_DEFAULT_IOCHARSET`). Il n'y a aucun moyen de spécifier ces paramètres pour les systèmes de fichiers ntfs au moment de la compilation du noyau.

10.3. Linux-6.16.1

Le paquet Linux contient le noyau Linux.

Temps de construction approximatif: 0,4 - 32 SBU (en général environ 2,5 SBU)
Espace disque requis: 1.7 - 14 Go (en général environ 2,3 Go)

10.3.1. Installation du noyau

Construire le noyau implique un certain nombre d'étapes — configuration, compilation et installation. Pour connaître les autres méthodes que celle employée par ce livre pour configurer le noyau, lisez le fichier README contenu dans les sources du noyau.

Important

Construire le noyau linux pour la première fois est l'une des tâches les plus difficiles de LFS. La bonne exécution de cette tâche dépend du matériel spécifique pour le système cible et de vos besoins spécifiques. Il y a plus de 12 000 entrées de configuration disponibles pour le noyau bien que seul un tiers d'entre elles soient requises pour la plupart des ordinateurs. Si vous n'êtes pas familiers de ce processus, les rédacteurs de LFS recommandent de suivre les procédure ci-dessous sans trop vous en écarter. L'objectif est d'avoir un premier système auquel vous pourrez vous connecter depuis la ligne de commande lorsque vous redémarrerez plus tard au Section 11.3, « Redémarrer le système. » Pour le moment, l'optimisation et la personnalisation ne sont pas des objectifs prioritaires.

Pour des informations d'ordre général sur la configuration du noyau, consultez *http://www.fr.linuxfromscratch.org/view/astuces/kernel-configuration-fr.txt*. Vous pouvez trouver des informations supplémentaires sur la configuration et la construction du noyau sur *http://www.kroah.com/lkn/*. Ces références sont un peu vieilles, mais donnent toujours un bon aperçu du processus.

Si tout le reste échoue, vous pouvez demander de l'aide sur la liste de diffusion *lfs-support*. Remarquez qu'il est nécessaire de s'enregistrer sur la liste. Cela permet d'éviter le spam.

Préparez la compilation en exécutant la commande suivante :

```
make mrproper
```

Ceci nous assure que le répertoire du noyau est propre. L'équipe du noyau recommande le lancement de cette commande avant chaque compilation du noyau. Vous ne devez pas supposer que le répertoire des sources est propre juste après avoir été déballé.

Il y a plusieurs manière de configurer les options du noyau. Habituellement, à travers une interface à menus, par exemple :

```
make menuconfig
```

Voici la signification des variables d'environnement facultatives de make :

LANG=<valeur_LANG_de_l_hote> LC_ALL=

Ceci rend identique les paramétrages régionaux à ceux utilisés sur l'hôte. C'est indispensable pour que l'interface ncurses de menuconfig soit correctement dessinée sur la console texte de Linux en UTF-8.

Assurez-vous si besoin de remplacer *<valeur_LANG_de_l_hote>* par la valeur de la variable $LANG de votre hôte. Vous pouvez utiliser à la place les valeurs $LC_ALL ou $LC_CTYPE de l'hôte.

make menuconfig

Cela lance une interface à menus en ncurses. Pour d'autres interfaces (graphiques), tapez **make help**.

Note

Un bon point de départ pour effectuer la configuration du noyau est de lancer **make defconfig**. Cela opérera une configuration de base de bonne qualité en prenant en compte l'architecture actuelle de votre système.

Assurez-vous d'activer, désactiver ou indiquer les fonctionnalités suivantes ou le système ne démarrera pas correctement voir pas du tout :

```
General setup --->
  [ ] Compile the kernel with warnings as errors                        [WERROR]
  CPU/Task time and stats accounting --->
    [*] Pressure stall information tracking                               [PSI]
    [ ]   Require boot parameter to enable pressure stall information tracking
                                                        ...  [PSI_DEFAULT_DISABLED]
  < > Enable kernel headers through /sys/kernel/kheaders.tar.xz      [IKHEADERS]
  [*] Control Group support --->                                       [CGROUPS]
    [*] Memory controller                                                [MEMCG]
  [ ] Configure standard kernel features (expert users) --->           [EXPERT]

Processor type and features --->
  [*] Build a relocatable kernel                                   [RELOCATABLE]
  [*]   Randomize the address of the kernel image (KASLR)       [RANDOMIZE_BASE]

General architecture-dependent options --->
  [*] Stack Protector buffer overflow detection                 [STACKPROTECTOR]
  [*]   Strong Stack Protector                             [STACKPROTECTOR_STRONG]

Device Drivers --->
  Generic Driver Options --->
    [ ] Support for uevent helper                                 [UEVENT_HELPER]
    [*] Maintain a devtmpfs filesystem to mount at /dev                [DEVTMPFS]
    [*]   Automount devtmpfs at /dev, after the kernel mounted the rootfs
                                                          ...  [DEVTMPFS_MOUNT]
  Firmware Drivers --->
    [*] Mark VGA/VBE/EFI FB as generic system framebuffer       [SYSFB_SIMPLEFB]
  Graphics support --->
    <*>    Direct Rendering Manager (XFree86 4.1.0 and higher DRI support) --->
                                                                    ...  [DRM]
    [*]    Display a user-friendly message when a kernel panic occurs
                                                                ...  [DRM_PANIC]
    (kmsg)   Panic screen formatter                          [DRM_PANIC_SCREEN]
    Supported DRM clients --->
      [*] Enable legacy fbdev support for your modesetting driver
                                                        ...  [DRM_FBDEV_EMULATION]
    Drivers for system framebuffers --->
      <*> Simple framebuffer driver                               [DRM_SIMPLEDRM]
    Console display driver support --->
      [*] Framebuffer Console support                        [FRAMEBUFFER_CONSOLE]
```

Activez certaines fonctionnalités supplémentaires si vous construisez un système 64 bits. Si vous utilisez menuconfig, activez-les dans l'ordre : d'abord *CONFIG_PCI_MSI*, puis *CONFIG_IRQ_REMAP* et enfin *CONFIG_X86_X2APIC* car une option ne s'affiche qu'après avoir sélectionné ses dépendances.

```
Processor type and features --->
  [*] x2APIC interrupt controller architecture support             [X86_X2APIC]

Device Drivers --->
  [*] PCI support --->                                                    [PCI]
    [*] Message Signaled Interrupts (MSI and MSI-X)                    [PCI_MSI]
  [*] IOMMU Hardware Support --->                                  [IOMMU_SUPPORT]
    [*] Support for Interrupt Remapping                               [IRQ_REMAP]
```

Si la partition du système LFS est un SSD NVME (c.-à-d. que le nœud de périphérique de la partition est `/dev/nvme*` au lieu de `/dev/sd*`), activez la prise en charge NVME ou le système LFS ne démarrera pas :

```
Device Drivers --->
  NVME Support --->
    <*> NVM Express block device                                    [BLK_DEV_NVME]
```

Vous pourriez souhaiter d'autres options selon les besoins de votre système. Pour une liste des options nécessaires pour les paquets BLFS, voir *L'index des options du noyau pour BLFS*.

Note

Si votre matériel hôte utilise UEFI et que vous souhaitez démarrer LFS sans, vous devrez ajuster certaines configurations du noyau en suivant *la page BLFS* **même si vous allez utiliser le chargeur d'amorçage UEFI de la distribution hôte**.

Voici pourquoi on vise les éléments de configuration ci-dessus :

`Randomize the address of the kernel image (KASLR)`
Active l'ASLR pour l'image du noyau, pour éviter certaines attaques basées sur les adresses fixes de données sensible ou de code dans le noyau.

`Compile the kernel with warnings as errors`
Cela peut causer un échec à la construction si le compilateur ou la configuration diffère de ceux des développeurs du noyau.

`Enable kernel headers through /sys/kernel/kheaders.tar.xz`
Cela demandera **cpio** pour construire le noyau. **cpio** n'est pas installé par LFS.

`Configure standard kernel features (expert users)`
Cette option affichera des options dans l'interface mais les changer peut s'avérer dangereux. N'utilisez pas cette option sauf si vous savez ce que vous faites.

`Strong Stack Protector`
Active SSP pour le noyau. Nous l'avons activée pour l'intégralité de l'espace utilisateur avec *`--enable-default-ssp`* en configurant GCC, mais le noyau n'utilise pas le paramètre GCC par défaut pour SSP. Nous l'activons ici explicitement.

`Support for uevent helper`
L'activation de cette option peut interférer avec la gestion de périphériques quand on utilise Udev.

`Maintain a devtmpfs`
Ceci créera des nœuds de périphérique automatiquement, générés par le noyau même sans Udev. Udev fonctionne alors sur cette base pour gérer les droits et l'ajout de liens symboliques. Cet élément de configuration est nécessaire pour tous les utilisateurs d'udev.

`Automount devtmpfs at /dev`
Cela montera la vue des périphériques du noyau sur /dev au changement de système de fichiers racine juste avant de charger l'init.

`Display a user-friendly message when a kernel panic occurs`
Cela fera afficher le message correctement dans le cas d'une panique du noyau et que le pilote DRM le prend en charge. Sans cela, il serait plus difficile de diagnostiquer les paniques : si aucun pilote DRM n'est utilisé, on est sur la console VGA qui ne peut contenir que 24 lignes et le message pertinent du noyau est souvent remplacé par le texte suivant. Si un pilote DRM est utilisé, l'affichage est souvent complètement désordonné lors d'une

panique. Depuis Linux-6.12, aucun des pilote dédiés aux modèles de GPU courants ne le prennent vraiment en charge, mais c'est pris en charge par le « pilote framebuffer simple » qui tourne sur le framebuffer VESA (ou EFI) avant le chargement du pilote GPU dédié. Si le pilote GPU dédié est construit en tant que module (au lieu de faire partie de l'image du noyau) et qu'aucun initramfs n'est utilisé, cette fonctionnalité fonctionnera correctement avant le montage du système de fichiers racine et c'est déjà suffisant pour fournir des informations sur les erreurs de configuration LFS qui causent une panique (par exemple un mauvais paramètre *root=* dans Section 10.4, « Utiliser GRUB pour paramétrer le processus de démarrage »).

Panic screen formatter

Indiquez kmsg pour vous assurer que les dernières lignes des messages du noyau sont affichées dans le cas d'une panique du noyau. La valeur par défaut, user, ne lui ferait afficher qu'un message de panique « convivial » qui n'est pas utile pour le diagnostic. La troisième possibilité, qr_code, ferait compresser les dernières lignes de messages en un code QR et l'afficher. Le code QR peut contenir plus de lignes que le texte brut et il peut être décodé par un appareil externe (comme un smartphone). Mais il nécessite le compilateur Rust que LFS ne fournit pas.

Mark VGA/VBE/EFI FB as generic system framebuffer et *Simple framebuffer driver*

Ils permettent d'utiliser le framebuffer VESA (ou le framebuffer EFI si vous démarrez le système LFS via UEFI) comme un périphérique DRM. Le framebuffer VESA sera configuré par GRUB (ou le framebuffer EFI sera configuré par le micrologiciel UEFI), donc le gestionnaire de panique DRM peut fonctionner avant que le pilote DRM spécifique au GPU ne soit chargé.

Enable legacy fbdev support for your modesetting driver et *Framebuffer Console support*

Ces options sont requises pour afficher la console Linux sur un GPU piloté par un pilote DRI (Infrastructure de Rendu Direct). Comme CONFIG_DRM (Gestionnaire de Rendu Direct) est activé, vous devriez également activer ces deux options ou vous verrez un écran noir une fois le pilote DRI chargé.

Support x2apic

Prend en charge le contrôleur d'interruption des processeurs x86 64 bits dans le mode x2APIC. x2APIC peut être activé par le micrologiciel sur les systèmes x86 64 bits, et un noyau sans cette option paniquera au démarrage si x2APIC est activé par le micrologiciel. Cette option n'a aucun effet mais ne cause aucun problème non plus si x2APIC est désactivé par le micrologiciel.

Sinon, **make oldconfig** peut être plus approprié dans certaines situations. Voir le fichier README pour plus d'informations.

Si vous le désirez, vous pouvez sauter la configuration du noyau en copiant le fichier de configuration, .config, du système hôte (en supposant qu'il est disponible) dans le répertoire linux-6.16.1 tout juste déballé. Néanmoins, nous ne recommandons pas cette option. Il est souvent meilleur d'explorer tous les menus de configuration et de créer la configuration du noyau à partir de zéro.

Compilez l'image du noyau et les modules :

```
make
```

Si vous utilisez des modules du noyau, il peut être nécessaire de les configurer dans le fichier /etc/modprobe.d. Des informations au sujet de la configuration du noyau et des modules se trouvent à la Section 9.3, « Manipulation des périphériques et modules » et dans le répertoire linux-6.16.1/Documentation de la documentation du noyau. Enfin, *modprobe.d(5)* pourrait aussi être intéressant.

À moins d'avoir désactivé la prise en charge des modules dans la configuration du noyau, installez les modules :

```
make modules_install
```

Une fois la compilation du noyau terminée, des étapes supplémentaires sont encore nécessaires pour terminer l'installation. Certains fichiers ont besoin d'être copiés dans le répertoire /boot.

Attention

Si vous avez décidé d'utliiser une partition `/boot` séparée pour le système LFS (en partageant éventuellement une partition `/boot` avec la distribution hôte), les fichiers copiés ci-dessous devraient aller là. La manière la plus simple de procéder est de lier /boot sur l'hôte (en dehors du chroot) à /mnt/lfs/boot avant de continuer. En tant qu'utilisateur `root` sur le *système hôte* :

```
mount /boot
```

Le chemin vers le nœud de périphérique est omis dans la commande car **mount** peut le lire dans `/etc/fstab`.

Le chemin vers l'image du noyau pourrait varier suivant la plateforme utilisée. Vous pouvez changer le nom du fichier ci-dessous selon votre goût, mais la nomenclature du nom de fichier devrait ressembler à *vmlinuz* pour être compatible avec le paramétrage automatique du processus de démarrage décrit dans la section à venir. La commande suivante présuppose une architecture x86 :

```
cp -iv arch/x86/boot/bzImage /boot/vmlinuz-6.16.1-lfs-12.4
```

`System.map` est un fichier de symboles pour le noyau. Il cartographie les points d'entrée de chaque fonction dans l'API du noyau, ainsi que les adresses de ses structures de données pendant l'exécution. Il sert de référence lors des investigations sur les problèmes de noyau. Lancez la commande suivante pour installer le fichier de symboles :

```
cp -iv System.map /boot/System.map-6.16.1
```

Le fichier de configuration du noyau `.config` produit à l'étape **make menuconfig** ci-dessus contient toutes les options de configuration choisies pour le noyau qui vient d'être compilé. Conserver ce fichier est une bonne idée pour pouvoir s'y référer plus tard :

```
cp -iv .config /boot/config-6.16.1
```

Installez la documentation du noyau Linux :

```
cp -r Documentation -T /usr/share/doc/linux-6.16.1
```

Il est important de noter que les fichiers dans le répertoire des sources du noyau n'appartiennent pas à *root*. Chaque fois qu'un paquet est déballé par l'utilisateur *root* (comme on a fait dans chroot), les fichiers ont les ID de l'utilisateur et du groupe de l'empaqueteur sur son système hôte. En principe ce n'est pas un problème car l'arborescence des sources est supprimée après l'installation. En revanche, l'arborescence de Linux est souvent conservée longtemps. Du coup, il y a des chances que tout ce que l'ID de l'utilisateur ayant déballé le paquet a utilisé ne soit affecté à quelqu'un d'autre sur la machine. Cette personne pourrait alors avoir un droit d'écriture sur les sources du noyau.

Note

Dans de nombreux cas, la configuration du noyau aura besoin d'être mise à jour pour les paquets qui serojnt installés plutard dans BLFS. Contrairement aux autres paquets, il n'est pas nécessaire de supprimer les sources du noyau après l'installation du noyau nouvellement construit.

Si vous conservez l'arborescence des sources du noyau, lancez **chown -R 0:0** sur le répertoire `linux-6.16.1` pour vous assurer que tous les fichiers appartiennent à *root*.

Si vous mettez à jour la configuration et reconstruisez le noyau à partir d'une arborescence des sources résultant d'une précédente compilation, vous ne devriez normalement **pas** avoir à lancer la commande **make mrproper**. La commande supprimerait le fichier `.config` et tous les fichiers `.o` de la construction précédente. Bien qu'il soit facile de restaurer un `.config` à partir de la copie dans `/boot`, supprimer tous les fichiers `.o` est une pure perte : pour un changement de configuration simple, souvent seuls quelques fichiers `.o` devront être reconstruits et le système de construction du noyau passera correctement les autres fichiers `.o` s'ils ne sont pas supprimés.

D'un autre côté, si vous avez mis à jour GCC, vous devriez exécuter **make clean** pour nettoyer tous les fichiers `.o` de la construction précédente, ou la nouvelle construction pourrait échouer.

Avertissement

Certaines documentations du noyau recommandent de créer un lien symbolique à partir de `/usr/src/linux` pointant vers le répertoire des sources du noyau. Ceci est spécifique aux noyaux antérieurs à la série 2.6 et *ne doit pas* être réalisé sur un système LFS car il peut poser des problèmes pour les paquets que vous souhaitez construire une fois votre système LFS de base complet.

10.3.2. Configuration de l'ordre de chargement des modules Linux

La plupart du temps, les modules Linux sont chargés automatiquement, mais il faut parfois des directives supplémentaires. Le programme qui charge les modules, **modprobe** ou **insmod**, utilise `/etc/modprobe.d/usb.conf` à cette fin. Il faut créer ce fichier afin que, si les pilotes USB (ehci_hcd, ohci_hcd et uhci_hcd) ont été construits en module, ils soient chargés dans le bon ordre ; ehci_hcd doit être chargé avant ohci_hcd et uhci_hcd afin d'éviter un avertissement au moment du démarrage.

Créez un nouveau `/etc/modprobe.d/usb.conf` en lançant ce qui suit :

```
install -v -m755 -d /etc/modprobe.d
cat > /etc/modprobe.d/usb.conf << "EOF"
# Début de /etc/modprobe.d/usb.conf

install ohci_hcd /sbin/modprobe ehci_hcd ; /sbin/modprobe -i ohci_hcd ; true
install uhci_hcd /sbin/modprobe ehci_hcd ; /sbin/modprobe -i uhci_hcd ; true

# Fin de /etc/modprobe.d/usb.conf
EOF
```

10.3.3. Contenu de Linux

Fichiers installés: config-6.16.1, vmlinuz-6.16.1-lfs-12.4 et System.map-6.16.1
Répertoires installés: /lib/modules, /usr/share/doc/linux-6.16.1

Descriptions courtes

`config-6.16.1` Contient toutes les options de configuration choisies pour le noyau

`vmlinuz-6.16.1-lfs-12.4` Le moteur du système Linux. Au démarrage de l'ordinateur, le noyau est la première partie du système d'exploitation à être chargée. Il détecte et initialise

tous composants matériels de l'ordinateur, puis rend disponible les composants dans une arborescence de fichiers pour les logiciels qui en ont besoin, et transforme une machine monoprocesseur en une machine multitâche capable d'exécuter plusieurs programmes quasi simultanément

`System.map-6.16.1` — Une liste d'adresses et de symboles donnant la correspondance entre les points d'entrée, et les adresses de toutes les fonctions et structures de données du noyau

10.4. Utiliser GRUB pour paramétrer le processus de démarrage

Note

Si votre système prend en charge l'UEFI et que vous souhaitez démarrer LFS avec l'UEFI, ignorez les instructions de cette page mais apprenez quand même la syntaxe de `grub.cfg` et la manière de spécifier une partition dans le fichier de cette page et configurez GRUB avec la prise en charge de l'UEFI en suivant les instructions de *la page BLFS*.

10.4.1. Introduction

Avertissement

Une mauvaise configuration de GRUB peut rendre votre système inutilisable si vous n'avez pas d'autre périphérique d'amorçage comme un cédérom. Cette section n'est pas obligatoire pour démarrer votre système LFS. Il se peut que vous vouliez simplement modifier votre chargeur de démarrage actuel, comme Grub-Legacy, GRUB2 ou LILO.

Assurez-vous d'avoir un disque de démarrage de façon à pouvoir « dépanner » l'ordinateur si celui-ci devenait inutilisable (non amorçable). Si vous n'avez pas déjà de périphérique de démarrage, vous pouvez en créer un. Afin que la procédure ci-dessous fonctionne, vous devez faire un tour du côté de BLFS et installer **`xorriso`** qui est dans le paquet *libisoburn*.

```
cd /tmp
grub-mkrescue --output=grub-img.iso
xorriso -as cdrecord -v dev=/dev/cdrw blank=as_needed grub-img.iso
```

10.4.2. Conventions de nommage de GRUB

GRUB utilise son propre système de dénomination des disques et des partitions. Il prend la forme *(hdn,m)* : *n* indique le numéro du disque dur et *m* le numéro de la partition. Le numéro du disque dur commence à partir de zéro, mais le numéro de la partition commence à partir d'un pour les partitions normales et à partir de cinq pour les partitions étendues. Ceci diffère des versions précédentes où les deux numéros commençaient à partir de zéro. Par exemple, la partition `sda1` correspond à *(hd0,1)* pour GRUB et la partition `sdb3` correspond à *(hd1,3)*. Contrairement à Linux, GRUB ne considère pas les lecteurs de CD-ROM comme des disques durs. Par exemple, si un CD se trouve sur `hdb` et un second disque dur sur `hdc`, ce dernier disque sera malgré tout *(hd1)*.

10.4.3. Réglage de la configuration

GRUB fonctionne en écrivant les données sur le premier secteur physique du disque dur. Ce secteur ne fait partie d'aucun système de fichiers. Les programmes accèdent alors aux modules de GRUB dans la partition de démarrage. L'emplacement par défaut est /boot/grub/.

L'emplacement de la partition de démarrage est un choix de l'utilisateur qui conditionne la configuration. L'utilisation d'une petite partition distincte (la taille suggérée est de 200 Mo) pour les informations de démarrage est recommandée. De cette façon, chaque construction, que ce soit LFS ou d'autres distributions commerciales, peut accéder aux mêmes fichiers de démarrage et n'importe quel système démarré peut y accéder. Si vous choisissez cette option, vous aurez besoin de monter la partition séparément, de déplacer tous les fichiers du répertoire `/boot` actuel (par exemple, le noyau linux que avez construit à l'étape précédente) vers la nouvelle partition. Vous aurez ensuite besoin de démonter la partition puis de la remonter en tant que `/boot`. Si vous procédez à ce changement, assurez-vous de mettre à jour `/etc/fstab`.

Il est possible de laisser la partition LFS actuelle dans le répertoire `/boot`, mais cela rendra la configuration de plusieurs systèmes plus difficile.

En utilisant les informations ci-dessus, déterminez le nom adapté à la partition racine (ou partition de démarrage, s'il en existe une distincte). Pour l'exemple suivant, supposons que la partition racine (ou la partition de démarrage) est `sda2`.

Installez les fichiers de GRUB dans `/boot/grub` et paramétrez le secteur d'amorçage :

Avertissement

La commande suivante va écraser le chargeur de démarrage actuel. Ne lancez pas la commande si ce n'est pas ce que vous désirez, par exemple si vous utilisez un gestionnaire de démarrage extérieur pour gérer le Master Boot Record (MBR).

```
grub-install /dev/sda
```

Note

Si le système a été démarré en UEFI, **grub-install** essayera d'installer des fichiers pour la cible *x86_64-efi*, mais ces fichiers n'ont pas été installés au Chapitre 8. Si c'est le cas, ajoutez `--target i386-pc` à la commande ci-dessus.

10.4.4. Créer le fichier de configuration de GRUB

Générez `/boot/grub/grub.cfg` :

```
cat > /boot/grub/grub.cfg << "EOF"
# Début de /boot/grub/grub.cfg
set default=0
set timeout=5

insmod part_gpt
insmod ext2
set root=(hd0,2)
set gfxpayload=1024x768x32

menuentry "GNU/Linux, Linux 6.16.1-lfs-12.4" {
        linux   /boot/vmlinuz-6.16.1-lfs-12.4 root=/dev/sda2 ro
}
EOF
```

Les commandes **insmod** chargent les modules GRUB nommés `part_gpt` et `ext2`. Malgré son nom, `ext2` prend en fait en charge les systèmes de fichiers `ext2`, `ext3` et `ext4`. La commande **grub-install** a intégré certains modules dans l'image principale de GRUB (installée dans le MBR ou la partition BIOS GRUB) pour accéder aux autres modules (dans `/boot/grub/i386-pc`) sans problème de type « l'œuf ou la poule », donc avec une configuration typique, ces deux modules sont déjà intégrés et ces deux commandes **insmod** ne feront rien. Mais elles ne posent aucun problème dans tous les cas, et peuvent être requises dans certaines configurations un peu rares.

La commande **set gfxpayload=1024x768x32** paramètre la résolution et la profondeur des couleurs du framebuffer VESA à passer au noyau. Elle est nécessaire pour que le pilote SimpleDRM du noyau utilise le framebuffer VESA. Vous pouvez utiliser une résolution ou une profondeur des couleurs différentes qui correspondent mieux à votre écran.

Note

Du point de vue de GRUB, les fichiers du noyau sont relatifs à la partition utilisée. Si vous avez utilisé une partition /boot distincte, supprimez /boot de la ligne *linux* ci-dessus. Vous devrez aussi modifier la ligne *set root* pour pointer vers la partition d'amorçage.

Note

La désignation GRUB pour une partition peut changer si vous ajoutez ou retirez des disques (dont les disques amovibles comme les clés USB). Le changement peut causer des échecs de démarrage parce que `grub.cfg` se réfère à « d'anciennes » désignations. Pour éviter ce problème, vous pouvez utiliser l'UUID de la partition et du système de fichiers pour indiquer une partition plutôt que d'utiliser la désignation GRUB. Exécutez **lsblk -o UUID,PARTUUID,PATH,MOUNTPOINT** pour afficher l'UUID de vos systèmes de fichiers (dans la colonne `UUID`) et de vos partitions (dans la colonne `PARTUUID`). Remplacez ensuite `set root=(hdx,y)` par `search --set=root --fs-uuid <UUID du système de fichiers où le noyau est installé>` et remplacez `root=/dev/sda2` par `root=PARTUUID=<UUID de la partition où LFS est construit>`.

L'UUID d'une partition et l'UUID du système de fichiers dans cette partition sont complètement différents. Certaines ressources en ligne peuvent vous indiquer d'utiliser `root=UUID=<UUID du système de fichiers>` au lieu de `root=PARTUUID=<UUID de la partition>`, mais cela nécessitera un initramfs qui va au-delà des objectifs de LFS.

Le nom du nœud de périphérique pour une partition dans `/dev` peut aussi changer (plus rarement que les désignations GRUB cependant). Vous pouvez aussi remplacer les chemins vers les nœuds de périphériques comme `/dev/sda1` par `PARTUUID=<UUID de la partition>` dans `/etc/fstab` pour éviter un échec au démarrage éventuel dans le cas où le nom du nœud de périphérique aurait changé.

GRUB est un programme extrêmement puissant et il offre un très grand nombre d'options pour démarrer depuis une large gamme de périphériques, de systèmes d'exploitation et de types de partition. Il a aussi beaucoup d'options de personnalisation telles que les écrans d'accueil graphiques, les annonces sonores, l'entrée à la souris, etc. Les détails de ces options vont au-delà des objectifs de cette introduction.

Attention

Il existe une commande, grub-mkconfig qui peut écrire automatiquement un fichier de configuration. Elle utilise un ensemble de scripts situés dans /etc/grub.d/ et elle détruira les personnalisations que vous aurez faites. Ces scripts sont d'abord conçus pour des distributions qui ne se basent pas sur les sources et ils ne sont pas recommandés pour LFS. Si vous installez une distribution Linux commerciale, il est fort probable que ce programme soit lancé. Assurez-vous de sauvegarder votre fichier grub.cfg.

Chapitre 11. La Fin

11.1. La Fin

Bien joué ! Le nouveau système LFS est installé ! Nous vous souhaitons de bien vous amuser avec votre tout nouveau système Linux fabriqué sur mesure.

Il est recommandé de créer le fichier `/etc/lfs-release`. Avec ce fichier, il vous est très facile (ainsi que pour nous si vous avez besoin de demander de l'aide) de savoir quelle version de LFS vous avez installé sur votre système. Créez ce fichier en lançant :

```
echo 12.4 > /etc/lfs-release
```

Certains paquets que vous installerez sur votre système pourront utiliser deux fichiers décrivant le système installé, soit sous forme binaire, soit à la construction.

Le premier affiche l'état de votre nouveau système, en respectant la Linux Standards Base (LSB). Pour créer ce fichier, lancez :

```
cat > /etc/lsb-release << "EOF"
DISTRIB_ID="Linux From Scratch"
DISTRIB_RELEASE="12.4"
DISTRIB_CODENAME="<votre nom ici>"
DISTRIB_DESCRIPTION="Linux From Scratch"
EOF
```

Le deuxième contient à peu près les mêmes informations et est utilisé par systemd et certains environnements de bureau. Pour créer ce fichier, lancez :

```
cat > /etc/os-release << "EOF"
NAME="Linux From Scratch"
VERSION="12.4"
ID=lfs
PRETTY_NAME="Linux From Scratch 12.4"
VERSION_CODENAME="<votre nom ici>"
HOME_URL="https://www.linuxfromscratch.org/lfs/"
RELEASE_TYPE="stable"
EOF
```

Assurez-vous de personnaliser les champs « DISTRIB_CODENAME » et « VERSION_CODENAME » pour que ce système ne soit que le vôtre.

11.2. Enregistrez-vous

Maintenant que vous avez terminé le livre, voulez-vous être enregistré comme utilisateur de LFS ? Allez directement sur *https://www.linuxfromscratch.org/cgi-bin/lfscounter.php* et enregistrez-vous comme utilisateur LFS en entrant votre nom et la première version de LFS que vous ayez utilisée.

Redémarrons sur LFS maintenant.

11.3. Redémarrer le système

Tous les logiciels sont à présent installés. Mais avant de redémarrer votre ordinateur, il y a encore quelques petites choses à vérifier :

- Si besoin, installez les *micrologiciels* nécessaires au bon fonctionnement du pilote de noyau de votre équipement.
- Assurez-vous qu'un mot de passe est initialisé pour l'utilisateur `root`.
- À ce stade, une relecture des fichiers de configuration suivants s'impose.

- /etc/fstab
- /etc/hosts
- /etc/inputrc
- /etc/profile
- /etc/resolv.conf
- /etc/vimrc
- /etc/sysconfig/ifconfig.eth0

Une fois cela fait, vous pouvez démarrer votre 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
mountpoint -q $LFS/dev/shm && umount -v $LFS/dev/shm
umount -v $LFS/dev
umount -v $LFS/run
umount -v $LFS/proc
umount -v $LFS/sys
```

Si plusieurs partitions ont été créées, démontez les autres partitions avant de démonter la principale en exécutant :

```
umount -v $LFS/home
umount -v $LFS
```

Démontez le système de fichiers LFS :

```
umount -v $LFS
```

Maintenant, redémarrez le système.

En supposant que le chargeur d'amorçage GRUB a été initialisé comme indiqué plus tôt, le menu est prêt à démarrer automatiquement *LFS 12.4*.

Une fois le redémarrage terminé, le système LFS est prêt à être utilisé. Ce que vous verrez est une simple invite « login: ». À partir de là, vous pouvez continuer avec *le livre BLFS* où vous trouverez plus de logiciels pour satisfaire vos besoins.

Si votre redémarrage ne fonctionne **pas**, il est l'heure de dépanner le système. Pour trouver des astuces sur les problèmes du premier démarrage, consultez *https://www.linuxfromscratch.org/lfs/troubleshooting.html*.

11.4. Ressources supplémentaires

Merci d'avoir lu le livre LFS. Nous espérons que vous avez trouvé ce livre utile et que vous en avez appris davantage sur le processus de création d'un système.

Maintenant que le système LFS est installé, vous êtes peut-être en train de vous demander « Et ensuite ? » Pour répondre à cette question, nous vous avons préparé une liste de ressources.

- Maintenance

  Les bogues et informations de sécurité sont rapportés régulièrement pour tous les logiciels. Comme un système LFS est compilé à partir des sources, c'est à vous de prendre en compte ces rapports. Il existe plusieurs ressources en ligne pour garder trace de tels rapports, quelques-unes d'entre elles sont indiquées ci-dessous :

  - *Informations de sécurité LFS*

    C'est une liste de vulnérabilités de sécurité découvertes dans le livre LFS après sa publication.

- *Open Source Security Mailing List*

  C'est une liste de diffusion pour discuter des failles de sécurité, des concepts et des pratiques de sécurité dans la communauté Open Source.

- Astuces LFS

  Les astuces LFS sont une collection de documents éducatifs soumis par des volontaires à la communauté LFS. Ces astuces sont disponibles sur *https://fr.linuxfromscratch.org/view/astuces/*.

- Listes de diffusion

  Il existe plusieurs listes de diffusion LFS auxquelles vous pouvez vous abonner si vous cherchez de l'aide, voulez suivre les derniers développements, voulez contribuer au projet et plus. Consultez le Chapitre 1 — Listes de diffusion pour plus d'informations.

- Le projet de documentation Linux (*The Linux Documentation Project*)

  Le projet de documentation Linux (LDP) a pour but de favoriser la collaboration concernant la documentation de Linux. Le LDP offre une large collection de guides pratiques, livres et pages de manuel. Il est disponible sur *https://fr.tldp.org/*.

11.5. Débuter After LFS

11.5.1. Décider que faire ensuite

Maintenant que LFS est terminé et que vous avez un système démarrable, que faire ? L'étape suivante est de décider comment l'utiliser. En général, il y a deux catégories génériques à prendre en compte : système de bureau ou serveur. Ces catégories ne sont pas mutuellement exclusives. Les applications requises pour chaque catégorie peuvent être combinées au sein d'un unique système, mais regardons les séparément pour le moment.

Un serveur est la catégorie la plus simple. En général elle consiste en un serveur web comme le *serveur HTTP Apache* et en un serveur de base de données comme *MariaDB*. Cependant d'autres services sont possibles. Le système d'exploitation inclus dans un appareil à but unique se trouve dans cette catégorie.

Un système de bureau, en revanche, est plus complexe car il requiert généralement un environnement utilisateur graphique tel que *LXDE*, *XFCE*, *KDE*, ou *Gnome* basé sur un *environnement graphique* de base et plusieurs applications à base graphique tel que *Firefox web browser*, *Thunderbird email client*, ou *LibreOffice office suite*. Ces applications requièrent encore plus de paquets (plusieurs centaines selon les capacités souhaitées) de prise en charge des applications et de bibliothèques.

Outre les aspects déjà mentionnés, il existe un ensemble d'applications d'administration système pour toutes sortes de systèmes. Ces applications sont dans le livre BLFS. Tous les paquets ne sont pas nécessaires dans tous les environnements. Par exemple *dhcpcd*, n'est pas approprié pour un serveur et *wireless_tools*, n'est utile que pour un système portable.

11.5.2. Travailler dans un environnement LFS de base

Lorsque vous vous lancez dans LFS, vous disposez de tous les outils internes pour construire des paquets supplémentaires. Malheureusement, l'environnement utilisateur est plutôt sommaire. Il y a plusieurs façons d'améliorer cela :

11.5.2.1. Travailler à partir du hôte LFS dans l’environnement chroot

Cette méthode permet de disposer d'un environnement graphique avec un navigateur intégral et des fonctions de copier/coller disponibles. Cette méthode permet d'utiliser des applications comme la version de l'hôte de wget pour télécharger les sources des paquets à un emplacement disponible lorsque l'on travaille dans l'environnement chroot.

Afin de construire correctement les paquets dans l'environnement chroot, montez les systèmes de fichiers virtuels s'ils ne sont pas déjà montés. Une façon de le faire est de créer un script sur le système **HOST** :

```
cat > ~/mount-virt.sh << "EOF"
#!/bin/bash

function mountbind
{
   if ! mountpoint $LFS/$1 >/dev/null; then
     $SUDO mount --bind /$1 $LFS/$1
     echo $LFS/$1 mounted
   else
     echo $LFS/$1 already mounted
   fi
}

function mounttype
{
   if ! mountpoint $LFS/$1 >/dev/null; then
     $SUDO mount -t $2 $3 $4 $5 $LFS/$1
     echo $LFS/$1 mounted
   else
     echo $LFS/$1 already mounted
   fi
}

if [ $EUID -ne 0 ]; then
  SUDO=sudo
else
  SUDO=""
fi

if [ x$LFS == x ]; then
  echo "LFS not set"
  exit 1
fi

mountbind dev
mounttype dev/pts devpts devpts -o gid=5,mode=620
mounttype proc    proc   proc
mounttype sys     sysfs  sysfs
mounttype run     tmpfs  run
if [ -h $LFS/dev/shm ]; then
  install -v -d -m 1777 $LFS$(realpath /dev/shm)
else
  mounttype dev/shm tmpfs tmpfs -o nosuid,nodev
fi

#mountbind usr/src
#mountbind boot
#mountbind home
EOF
```

Remarquez que les trois dernières commandes du script sont commentées. Elles sont utiles si ces répertoires sont montés comme des partitions séparées sur le système hôte et s'ils seront montés lors du démarrage du système LFS/BLFS finalisé.

Le script peut être exécuté avec **bash ~/mount-virt.sh** en tant qu'utilisateur normal (recommandé) ou en tant que `root`. S'il est exécuté en tant qu'utilisateur normal, sudo est requis dans le système hôte.

Un autre problème signalé par le script est où stocker les fichiers de paquets téléchargés. Cet emplacement est arbitraire. Il peut être dans le répertoire personnel d'un utilisateur ordinaire comme ~/sources ou dans un emplacement global comme /usr/src. Notre recommandation est de ne pas mélanger les sources BLFS et les sources LFS dans (à partir de l'environnement chroot) /sources. Dans tous les cas, les paquets doivent être accessibles à l'intérieur de l'environnement chroot.

Une dernière fonctionnalité pratique présentée ici permet de rationaliser le processus d'entrée dans l'environnement chroot. Cela peut être fait avec un alias placé dans le fichier ~/.bashrc d'un utilisateur sur le système hôte :

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

Cet alias est un peu délicat à cause des guillemets et des niveaux de barres obliques. Tout doit figurer sur une seule ligne. La commande ci-dessus a été divisée en deux pour des raisons de présentation.

11.5.2.2. Travailler à distance via ssh

Cette méthode offre également un environnement graphique complet, mais nécessite d'abord d'installer *sshd* sur le système LFS. Cette méthode requiert aussi un deuxième ordinateur. Cette méthode a l'avantage d'être simple en ne nécessitant pas la complexité de l'environnement chroot. Cette méthode utilise le noyau LFS déjà construit pour l'installation de paquets supplémentaires en fournissant tout de même un système complet pour l'installation des paquets.

Vous pouvez utiliser la commande **scp** pour téléverser les sources du paquet à construire sur le système LFS. Si vous voulez plutôt télécharger les sources sur le système LFS directement, installez *libtasn1*, *p11-kit*, *make-ca* et *wget* en chroot (ou envoyez leurs sources avec **scp** après avoir démarré le système LFS).

11.5.2.3. Travailler depuis la ligne de commande LFS

Cette méthode nécessite l'installation de *libtasn1*, *p11-kit*, *make-ca*, *wget*, *gpm*, et *links* (ou *lynx*) dans chroot et ensuite d'un redémarrage dans le nouveau système LFS. À ce stade, le système par défaut possède six consoles virtuelles. Changer de console est aussi simple que d'utiliser les combinaisons de touches **Alt**+**Fx** où **Fx** est entre **F1** et **F6**. Les combinaisons des touches **Alt**+$\leftarrow$ et **Alt**+$\rightarrow$ changeront aussi la console.

Maintenant, vous pouvez vous connecter à deux consoles virtuelles différentes et exécuter le navigateur web links ou lynx dans une console tandis que vous exécutez bash dans l'autre. GPM permet alors de copier les commandes du navigateur avec le bouton gauche de la souris, de changer de console et de les coller dans l'autre console.

Note

À titre d'information, le changement de console virtuelle peut également être effectué à partir d'une fenêtre X avec la combinaison de touches **Ctrl**+**Alt**+**Fx**, mais l'opération de copie de la souris ne fonctionne pas entre l'interface graphique et une console virtuelle. Vous pouvez revenir à l'affichage de la fenêtre X avec la combinaison **Ctrl**+**Alt**+**Fx**, où **Fx** est généralement **F1** mais peut être **F7**.

Partie V. Annexes

Annexe A. Acronymes et termes

ABI *Application Binary Interface* ou interface binaire-programme

ALFS *Automated Linux From Scratch*

API Interface de programmation d'application

ASCII *American Standard Code for Information Interchange* ou code américain normalisé pour l'échange d'information

BIOS *Basic Input/Output System* ou système d'entrées/sorties de base

BLFS *Beyond Linux From Scratch*

BSD *Berkeley Software Distribution*

chroot *change root*

CMOS *Complementary Metal Oxide Semiconductor* ou semi-conducteur à oxyde métallique complémentaire

COS *Class Of Service*

CPU *Central Processing Unit* ou unité centrale de traitement

CRC Contrôle de redondance cyclique

CVS *Concurrent Versions System* ou système de gestion de versions

DHCP *Dynamic Host Configuration Protocol* ou protocole de configuration dynamique d'adressage serveur

DNS *Domain Name Service* ou service de nom de domaine

EGA *Enhanced Graphics Adapter* ou adaptateur graphique amélioré

ELF Format exécutable et liable

EOF *End of File* ou fin de fichier

EQN Équation

ext2 *second extended file system*

ext3 *third extended file system*

ext4 *fourth extended file system*

FAQ Foire aux questions

FHS *Filesystem Hierarchy Standard* ou hiérarchie standard du système de fichiers

FIFO *First-In, First Out* ou premier arrivé, premier servi

FQDN *Fully Qualified Domain Name* ou nom de domaine pleinement qualifié

FTP *File Transfer Protocol* ou protocole de transfert de fichiers

Go Gigaoctet

GCC *GNU Compiler Collection*

GID Identificateur de groupe

GMT Temps moyen de Greenwich

HTML *Hypertext Markup Language*

IDE *Integrated Drive Electronics* ou gestion de périphériques à électronique intégrée

IEEE *Institute of Electrical and Electronic Engineers*

IO *Input/Output* ou Entrées/Sorties

IP Protocole Internet

IPC Communication inter-processus

IRC *Internet Relay Chat* ou service d'échanges textuels en temps réel

ISO *International Organization for Standardization*

ISP *Internet Service Provider* ou fournisseur d'accès Internet (FAI)

Ko Kilooctet

LED *Light Emitting Diode* ou diode électroluminescente

LFS *Linux From Scratch*

LSB *Linux Standard Base*

Mo Mégaoctet

MBR *Master Boot Record* ou secteur d'amorçage

MD5 *Message Digest 5*

NIC *Network Interface Card* ou carte d'interface réseau

NLS *Native Language Support* ou prise en charge de la langue maternelle

NNTP *Network News Transport Protocol* ou protocole de transfert UseNet

NPTL *Native POSIX Threading Library*

OSS *Open Sound System*

PCH *Pre-Compiled Headers*

PCRE *Perl Compatible Regular Expression*

PID Identificateur de processus

PTY Pseudo terminal

QOS *Quality Of Service* ou qualité de service

RAM *Random Access Memory* ou mémoire vive

RPC *Remote Procedure Call* ou appel de procédure distante

RTC *Real Time Clock* ou horloge temps réel

SBU *Standard Build Unit*

SCO *The Santa Cruz Operation*

SHA1 *Secure-Hash Algorithm 1*

TLDP Le projet de documentation Linux (*The Linux Documentation Project*)

TFTP *Trivial File Transfer Protocol* ou protocole simplifié de transfert de fichiers

TLS *Thread-Local Storage* ou mémoire locale de thread

UID Identificateur utilisateur

umask *user file-creation mask*

USB *Universal Serial Bus*

UTC Temps universel coordonné

UUID Identificateur universellement unique

VC Console Virtuelle

VGA Adaptateur graphique vidéo

VT Terminal virtuel

Annexe B. Remerciements

Nous aimerions remercier les personnes et organisations suivantes pour leurs contributions au projet Linux From Scratch.

- *Gerard Beekmans* <gerard@linuxfromscratch.org> – Créateur de LFS
- *Bruce Dubbs* <bdubbs@linuxfromscratch.org> – Rédacteur en chef de LFS
- *Jim Gifford* <jim@linuxfromscratch.org> – Co-Leader du projet CLFS
- *Pierre Labastie* <pierre@linuxfromscratch.org> – Rédacteur de BLFS et meneur de ALFS
- *DJ Lucas* <dj@linuxfromscratch.org> – Rédacteur de LFS et de BLFS
- *Ken Moffat* <ken@linuxfromscratch.org> – Rédacteur de BLFS
- Sans compter les autres personnes sur les diverses listes de diffusion de LFS et BLFS qui ont aidé à rendre possible ce livre par leurs suggestions, leurs tests ; leurs soumissions de rapports de bogue, d'instructions et leurs retours d'expérience en installant divers paquets.

Traducteurs

- *Manuel Canales Esparcia* <macana@macana-es.com> – Projet de traduction de LFS en espagnol
- *Johan Lenglet* <johan@linuxfromscratch.org> – Projet de traduction de LFS en français jusqu'en 2008
- *Jean-Philippe Mengual* <jmengual@linuxfromscratch.org> – Projet de traduction de LFS en français de 2008 à 2016
- *Julien Lepiller* <jlepiller@linuxfromscratch.org> – Projet de traduction de LFS en français depuis 2017
- *Anderson Lizardo* <lizardo@linuxfromscratch.org> – Projet de traduction de LFS en portugais historique
- *Jamenson Espindula* <jafesp@gmail.com> – Projet de traduction de LFS en portugais depuis 2022
- *Thomas Reitelbach* <tr@erdfunkstelle.de> – Projet de traduction de LFS en allemand

Mainteneurs de miroirs

Miroirs nord-américains

- *Scott Kveton* <scott@osuosl.org> – miroir lfs.oregonstate.edu
- *William Astle* <lost@l-w.net> – miroir ca.linuxfromscratch.org
- *Eujon Sellers* <jpolen@rackspace.com> – miroir lfs.introspeed.com
- *Justin Knierim* <tim@idge.net> – miroir lfs-matrix.net

Miroirs sud-américains

- *Manuel Canales Esparcia* <manuel@linuxfromscratch.org> – miroir lfsmirror.lfs-es.info
- *Luis Falcon* <Luis Falcon> – miroir torredehanoi.org

Miroirs européens

- *Guido Passet* <guido@primerelay.net> – miroir nl.linuxfromscratch.org
- *Bastiaan Jacques* <baafie@planet.nl> – miroir lfs.pagefault.net
- *Sven Cranshoff* <sven.cranshoff@lineo.be> – miroir lfs.lineo.be
- Scarlet Belgium – miroir lfs.scarlet.be

- *Sebastian Faulborn* <info@aliensoft.org> – miroir lfs.aliensoft.org
- *Stuart Fox* <stuart@dontuse.ms> – miroir lfs.dontuse.ms
- *Ralf Uhlemann* <admin@realhost.de> – miroir lfs.oss-mirror.org
- *Antonin Sprinzl* <Antonin.Sprinzl@tuwien.ac.at> – miroir at.linuxfromscratch.org
- *Fredrik Danerklint* <fredan-lfs@fredan.org> – miroir se.linuxfromscratch.org
- *Franck* <franck@linuxpourtous.com> – miroir lfs.linuxpourtous.com
- *Philippe Baque* <baque@cict.fr> – miroir lfs.cict.fr
- *Vitaly Chekasin* <gyouja@pilgrims.ru> – miroir lfs.pilgrims.ru
- *Benjamin Heil* <kontakt@wankoo.org> – miroir lfs.wankoo.org
- *Anton Maisak* <info@linuxfromscratch.org.ru> – miroir linuxfromscratch.org.ru

Miroirs asiatiques

- *Satit Phermsawang* <satit@wbac.ac.th> – miroir lfs.phayoune.org
- *Shizunet Co.,Ltd.* <info@shizu-net.jp> – miroir lfs.mirror.shizu-net.jp

Miroirs australiens

- *Jason Andrade* <jason@dstc.edu.au> – miroir au.linuxfromscratch.org

Anciens membres de l'équipe du projet

- *Christine Barczak* <theladyskye@linuxfromscratch.org> – Rédacteur du livre LFS
- Archaic <archaic@linuxfromscratch.org> – Rédacteur technique LFS, leader du projet HLFS, éditeur de BLFS, mainteneur des projets d'astuces et correctifs
- *Matthew Burgess* <matthew@linuxfromscratch.org> – Leader du projet LFS, rédacteur technique LFS/éditeur
- *Nathan Coulson* <nathan@linuxfromscratch.org> – Mainteneur de LFS-Bootscripts
- Timothy Bauscher
- Robert Briggs
- Ian Chilton
- *Jeroen Coumans* <jeroen@linuxfromscratch.org> – Développeur du site Web, mainteneur de la FAQ
- *Manuel Canales Esparcia* <manuel@linuxfromscratch.org> – Mainteneur de LFS/BLFS/HLFS en XML et XSL
- Alex Groenewoud – Rédacteur technique LFS
- Marc Heerdink
- *Jeremy Huntwork* <jhuntwork@linuxfromscratch.org> – Rédacteur technique LFS, mainteneur du LiveCD LFS
- *Bryan Kadzban* <bryan@linuxfromscratch.org> – Rédacteur technique LFS
- Mark Hymers
- Seth W. Klein – Mainteneur de la FAQ
- *Nicholas Leippe* <nicholas@linuxfromscratch.org> – Mainteneur du Wiki
- *Anderson Lizardo* <lizardo@linuxfromscratch.org> – Mainteneur des scripts d'arrière-plan du site Web
- *Randy McMurchy* <randy@linuxfromscratch.org> – Leader du projet BLFS, éditeur LFS

- *Dan Nicholson* <dnicholson@linuxfromscratch.org> – Rédacteur de LFS et BLFS
- *Alexander E. Patrakov* <alexander@linuxfromscratch.org> – Rédacteur Technique LFS, éditeur des traductions LFS, mainteneur du LiveCD LFS
- Simon Perreault
- *Scot Mc Pherson* <scot@linuxfromscratch.org> – Mainteneur de LFS NNTP Gateway
- *Douglas R. Reno* <renodr@linuxfromscratch.org> – Rédacteur Systemd
- *Ryan Oliver* <ryan@linuxfromscratch.org> – Co-Leader du projet CLFS
- *Greg Schafer* <gschafer@zip.com.au> – Rédacteur technique LFS et architecte de la nouvelle méthode de construction activant le 64 bits
- Jesse Tie-Ten-Quee – Rédacteur technique LFS
- *James Robertson* <jwrober@linuxfromscratch.org> – Mainteneur Bugzilla
- *Tushar Teredesai* <tushar@linuxfromscratch.org> – Rédacteur du livre BLFS, leader du projet d'astuces et correctifs
- *Jeremy Utley* <jeremy@linuxfromscratch.org> – Rédacteur technique LFS, mainteneur Bugzilla, mainteneur de LFS-Bootscripts
- *Zack Winkles* <zwinkles@gmail.com> – Rédacteur technique LFS

Annexe C. Dépendances

La bonne compilation et la bonne installation de chaque paquet compilé dans LFS dépend d'un ou de plusieurs autres paquets. Certains paquets participent même aux dépendances circulaires, c'est-à-dire que le premier paquet dépend du second qui dépend à son tour du premier. À cause de ces dépendances, l'ordre dans lequel les paquets sont compilés dans LFS est très important. Le but de cette page est de documenter les dépendances de chaque paquet compilé dans LFS.

Pour chaque paquet que nous compilons, nous avons listé trois, voire cinq types de dépendances. La première répertorie quels autres paquets doivent être disponibles afin de compiler et d'installer le paquet en question. La deuxième liste les paquets qui doivent être disponibles lorsqu'un programme ou une bibliothèque du paquet est exécuté. La troisième concerne les paquets qui, en plus de ceux de la première liste, doivent être disponibles afin de lancer les suites de test. La quatrième liste de dépendances contient les paquets qui exigent que ce paquet soit compilé et installé à l'emplacement final avant qu'ils ne soient compilés et installés.

La dernière liste de dépendances répertorie les paquets facultatifs qui ne sont pas destinées à LFS mais qui pourraient vous être utiles. Ces paquets peuvent avoir eux-mêmes des dépendances supplémentaires obligatoires ou facultatives. Pour ces dépendances, la pratique recommandée consiste à les installer après avoir terminé le livre LFS puis à revenir en arrière pour reconstruire le paquet LFS. Dans certains cas, la réinstallation est traitée dans BLFS.

Acl

L'installation dépend de: Attr, Bash, Binutils, Coreutils, GCC, Gettext, Grep, M4, Make, Perl, Sed et Texinfo

Requis à l'exécution: Attr et Glibc

La suite de tests dépend de: Automake, Diffutils, Findutils et Libtool

Doit être installé avant: Coreutils, Sed, Tar et Vim

Dépendances facultatives: Aucun

Attr

L'installation dépend de: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, M4, Make, Perl, Sed et Texinfo

Requis à l'exécution: Glibc

La suite de tests dépend de: Automake, Diffutils, Findutils et Libtool

Doit être installé avant: Acl, Libcap et Patch

Dépendances facultatives: Aucun

Autoconf

L'installation dépend de: Bash, Coreutils, Grep, M4, Make, Perl, Sed et Texinfo

Requis à l'exécution: Bash, Coreutils, Grep, M4, Make, Sed et Texinfo

La suite de tests dépend de: Automake, Diffutils, Findutils, GCC et Libtool

Doit être installé avant: Automake et Coreutils

Dépendances facultatives: *Emacs*

Automake

L'installation dépend de: Autoconf, Bash, Coreutils, Gettext, Grep, M4, Make, Perl, Sed et Texinfo
Requis à l'exécution: Bash, Coreutils, Grep, M4, Sed et Texinfo
La suite de tests dépend de: Binutils, Bison, Bzip2, DejaGNU, Diffutils, Expect, Findutils, Flex, GCC, Gettext, Gzip, Libtool et Tar
Doit être installé avant: Coreutils
Dépendances facultatives: Aucun

Bash

L'installation dépend de: Bash, Binutils, Bison, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Make, Ncurses, Patch, Readline, Sed et Texinfo
Requis à l'exécution: Glibc, Ncurses et Readline
La suite de tests dépend de: Expect et Shadow
Doit être installé avant: Aucun
Dépendances facultatives: *Xorg*

Bc

L'installation dépend de: Bash, Binutils, Coreutils, GCC, Glibc, Grep, Make et Readline
Requis à l'exécution: Glibc, Ncurses et Readline
La suite de tests dépend de: Gawk
Doit être installé avant: Linux
Dépendances facultatives: Aucun

Binutils

L'installation dépend de: Bash, Binutils, Coreutils, Diffutils, File, Flex, Gawk, GCC, Glibc, Grep, Make, Perl, Pkgconf, Sed, Texinfo, Zlib et Zstd
Requis à l'exécution: Glibc, Zlib et Zstd
La suite de tests dépend de: DejaGNU et Expect
Doit être installé avant: Aucun
Dépendances facultatives: *Elfutils* et *Jansson*

Bison

L'installation dépend de: Bash, Binutils, Coreutils, Diffutils, GCC, Gettext, Glibc, Grep, M4, Make, Perl et Sed
Requis à l'exécution: Glibc
La suite de tests dépend de: Diffutils, Findutils et Flex
Doit être installé avant: Kbd et Tar
Dépendances facultatives: *Doxygen*

Bzip2

L'installation dépend de: Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Make, et Patch
Requis à l'exécution: Glibc
La suite de tests dépend de: Aucun
Doit être installé avant: File et Libelf
Dépendances facultatives: Aucun

Coreutils

L'installation dépend de: Autoconf, Automake, Bash, Binutils, Coreutils, GCC, Gettext, Glibc, GMP, Grep, Libcap, Make, OpenSSL, Patch, Perl, Sed et Texinfo
Requis à l'exécution: Glibc
La suite de tests dépend de: Diffutils, E2fsprogs, Findutils, Shadow et Util-linux
Doit être installé avant: Bash, Diffutils, Findutils, Man-DB et Udev
Dépendances facultatives: *Expect.pm* et *IO::Tty*

DejaGNU

L'installation dépend de: Bash, Coreutils, Diffutils, Expect, GCC, Grep, Make, Sed et Texinfo
Requis à l'exécution: Expect et Bash
La suite de tests dépend de: Aucun
Doit être installé avant: Aucun
Dépendances facultatives: Aucun

Diffutils

L'installation dépend de: Bash, Binutils, Coreutils, Gawk, GCC, Gettext, Glibc, Grep, Make, Sed et Texinfo
Requis à l'exécution: Glibc
La suite de tests dépend de: Perl
Doit être installé avant: Aucun
Dépendances facultatives: Aucun

E2fsprogs

L'installation dépend de: Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Gzip, Make, Pkgconf, Sed, Texinfo et Util-linux
Requis à l'exécution: Glibc et Util-linux
La suite de tests dépend de: Procps-ng et Psmisc
Doit être installé avant: Aucun
Dépendances facultatives: Aucun

Expat

L'installation dépend de: Bash, Binutils, Coreutils, Gawk, GCC, Glibc, Grep, Make et Sed
Requis à l'exécution: Glibc
La suite de tests dépend de: Aucun
Doit être installé avant: Python et XML::Parser
Dépendances facultatives: Aucun

Expect

L'installation dépend de: Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Grep, Make, Patch, Sed et Tcl
Requis à l'exécution: Glibc et Tcl
La suite de tests dépend de: Aucun
Doit être installé avant: Aucun
Dépendances facultatives: *Tk*

File

L'installation dépend de: Bash, Binutils, Bzip2, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Make, Sed, Xz et Zlib
Requis à l'exécution: Glibc, Bzip2, Xz et Zlib
La suite de tests dépend de: Aucun
Doit être installé avant: Aucun
Dépendances facultatives: *libseccomp*

Findutils

L'installation dépend de: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, Make, Sed et Texinfo
Requis à l'exécution: Bash et Glibc
La suite de tests dépend de: DejaGNU, Diffutils et Expect
Doit être installé avant: Aucun
Dépendances facultatives: Aucun

Flex

L'installation dépend de: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, M4, Make, Patch, Sed et Texinfo
Requis à l'exécution: Bash, Glibc et M4
La suite de tests dépend de: Bison et Gawk
Doit être installé avant: Binutils, IProute2, Kbd, Kmod et Man-DB
Dépendances facultatives: Aucun

Flit-Core

L'installation dépend de: Python
Requis à l'exécution: Python
La suite de tests dépend de: Aucune suite de tests disponible
Doit être installé avant: Packaging et Wheel
Dépendances facultatives: *pytest* et *testpath*

Gawk

L'installation dépend de: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, GMP, Grep, Make, MPFR, Patch, Readline, Sed et Texinfo
Requis à l'exécution: Bash, Glibc et Mpfr
La suite de tests dépend de: Diffutils
Doit être installé avant: Aucun
Dépendances facultatives: *libsigsegv*

GCC

L'installation dépend de: Bash, Binutils, Coreutils, Diffutils, Findutils, Gawk, GCC, Gettext, Glibc, GMP, Grep, M4, Make, MPC, MPFR, Patch, Perl, Sed, Tar, Texinfo et Zstd
Requis à l'exécution: Bash, Binutils, Glibc, Mpc et Python
La suite de tests dépend de: DejaGNU, Expect et Shadow
Doit être installé avant: Aucun
Dépendances facultatives: *GDC*, *GNAT* et *ISL*

GDBM

L'installation dépend de: Bash, Binutils, Coreutils, Diffutils, GCC, Grep, Make et Sed
Requis à l'exécution: Bash, Glibc et Readline
La suite de tests dépend de: Aucun
Doit être installé avant: Aucun
Dépendances facultatives: Aucun

Gettext

L'installation dépend de: Bash, Binutils, Coreutils, Gawk, GCC, Glibc, Grep, Make, Ncurses, Sed et Texinfo
Requis à l'exécution: Acl, Bash, Gcc et Glibc
La suite de tests dépend de: Diffutils, Perl et Tcl
Doit être installé avant: Automake et Bison
Dépendances facultatives: *libunistring* et *libxml2*

Glibc

L'installation dépend de: Bash, Binutils, Bison, Coreutils, Diffutils, Gawk, GCC, Gettext, Grep, Gzip, Linux API Headers, Make, Perl, Python, Sed et Texinfo
Requis à l'exécution: Aucun
La suite de tests dépend de: File
Doit être installé avant: Aucun
Dépendances facultatives: Aucun

GMP

L'installation dépend de: Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, M4, Make, Sed et Texinfo
Requis à l'exécution: GCC et Glibc
La suite de tests dépend de: Aucun
Doit être installé avant: MPFR et GCC
Dépendances facultatives: Aucun

Gperf

L'installation dépend de: Bash, Binutils, Coreutils, GCC, Glibc et Make
Requis à l'exécution: GCC et Glibc
La suite de tests dépend de: Diffutils et Expect
Doit être installé avant: Aucun
Dépendances facultatives: Aucun

Grep

L'installation dépend de: Bash, Binutils, Coreutils, Diffutils, GCC, Gettext, Glibc, Grep, Make, Patch, Sed et Texinfo
Requis à l'exécution: Glibc
La suite de tests dépend de: Gawk
Doit être installé avant: Man-DB
Dépendances facultatives: *PCRE2* et *libsigsegv*

Groff

L'installation dépend de: Bash, Binutils, Bison, Coreutils, Gawk, GCC, Glibc, Grep, Make, Patch, Sed et Texinfo
Requis à l'exécution: GCC, Glibc et Perl
La suite de tests dépend de: Aucun
Doit être installé avant: Man-DB
Dépendances facultatives: *ghostscript* et *Uchardet*

GRUB

L'installation dépend de: Bash, Binutils, Bison, Coreutils, Diffutils, GCC, Gettext, Glibc, Grep, Make, Ncurses, Sed, Texinfo et Xz

Requis à l'exécution: Bash, GCC, Gettext, Glibc, Xz et Sed

La suite de tests dépend de: Aucun

Doit être installé avant: Aucun

Dépendances facultatives: Aucun

Gzip

L'installation dépend de: Bash, Binutils, Coreutils, GCC, Glibc, Grep, Make, Sed, et Texinfo

Requis à l'exécution: Bash et Glibc

La suite de tests dépend de: Diffutils et Less

Doit être installé avant: Man-DB

Dépendances facultatives: Aucun

Iana-Etc

L'installation dépend de: Coreutils

Requis à l'exécution: Aucun

La suite de tests dépend de: Aucune suite de tests disponible

Doit être installé avant: Perl

Dépendances facultatives: Aucun

Inetutils

L'installation dépend de: Bash, Binutils, Coreutils, GCC, Glibc, Grep, Make, Ncurses, Patch, Sed, Texinfo et Zlib

Requis à l'exécution: GCC, Glibc, Ncurses et Readline

La suite de tests dépend de: Aucun

Doit être installé avant: Tar

Dépendances facultatives: Aucun

Intltool

L'installation dépend de: Bash, Gawk, Glibc, Make, Perl, Sed et XML::Parser

Requis à l'exécution: Autoconf, Automake, Bash, Glibc, Grep, Perl et Sed

La suite de tests dépend de: Perl

Doit être installé avant: Aucun

Dépendances facultatives: Aucun

IProute2

L'installation dépend de: Bash, Bison, Coreutils, Flex, GCC, Glibc, Make, Libcap, Libelf, Linux API Headers, Pkgconf et Zlib
Requis à l'exécution: Bash, Coreutils, Glibc, Libcap, Libelf et Zlib
La suite de tests dépend de: Aucune suite de tests disponible
Doit être installé avant: Aucun
Dépendances facultatives: *Berkeley DB*, *iptables*, *libbpf*, *libmnl* et *libtirpc*

Jinja2

L'installation dépend de: MarkupSafe, Python, Setuptools et Wheel
Requis à l'exécution: MarkupSafe et Python
La suite de tests dépend de: Aucune suite de tests disponible
Doit être installé avant: Udev
Dépendances facultatives: Aucun

Kbd

L'installation dépend de: Bash, Binutils, Bison, Coreutils, Flex, GCC, Gettext, Glibc, Gzip, Make, Patch et Sed
Requis à l'exécution: Bash, Coreutils et Glibc
La suite de tests dépend de: Aucun
Doit être installé avant: Aucun
Dépendances facultatives: *Linux-PAM*

Kmod

L'installation dépend de: Bash, Binutils, Bison, Coreutils, Flex, GCC, Gettext, Glibc, Gzip, Make, OpenSSL, Pkgconf, Sed, Xz et Zlib
Requis à l'exécution: Glibc, Xz et Zlib
La suite de tests dépend de: Aucune suite de tests disponible
Doit être installé avant: Udev
Dépendances facultatives: *scdoc* (pour les pages de manuel)

Less

L'installation dépend de: Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Grep, Make, Ncurses et Sed
Requis à l'exécution: Glibc et Ncurses
La suite de tests dépend de: Aucun
Doit être installé avant: Gzip
Dépendances facultatives: *PCRE2* ou *PCRE*

Libcap

L'installation dépend de: Attr, Bash, Binutils, Coreutils, GCC, Glibc, Perl, Make et Sed
Requis à l'exécution: Glibc
La suite de tests dépend de: Aucun
Doit être installé avant: IProute2 et Shadow
Dépendances facultatives: *Linux-PAM*

Libelf

L'installation dépend de: Bash, Binutils, Bzip2, Coreutils, GCC, Glibc, Make, Xz, Zlib et Zstd
Requis à l'exécution: Bzip2, Glibc, Xz, Zlib et Zstd
La suite de tests dépend de: Aucun
Doit être installé avant: IProute2 et Linux
Dépendances facultatives: Aucun

Libffi

L'installation dépend de: Bash, Binutils, Coreutils, GCC, Glibc, Make et Sed
Requis à l'exécution: Glibc
La suite de tests dépend de: DejaGnu
Doit être installé avant: Python
Dépendances facultatives: Aucun

Libpipeline

L'installation dépend de: Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Make, Sed et Texinfo
Requis à l'exécution: Glibc
La suite de tests dépend de: Check et Pkgconf
Doit être installé avant: Man-DB
Dépendances facultatives: Aucun

Libtool

L'installation dépend de: Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Make, Sed et Texinfo
Requis à l'exécution: Autoconf, Automake, Bash, Binutils, Coreutils, File, GCC, Glibc, Grep, Make et Sed
La suite de tests dépend de: Autoconf, Automake et Findutils
Doit être installé avant: Aucun
Dépendances facultatives: Aucun

Libxcrypt

L'installation dépend de: Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Make, Perl et Sed
Requis à l'exécution: Glibc
La suite de tests dépend de: Aucun
Doit être installé avant: Perl, Python, Shadow et Udev
Dépendances facultatives: Aucun

Linux

L'installation dépend de: Bash, Bc, Binutils, Coreutils, Diffutils, Findutils, GCC, Glibc, Grep, Gzip, Kmod, Libelf, Make, Ncurses, OpenSSL, Perl et Sed
Requis à l'exécution: Aucun
La suite de tests dépend de: Aucune suite de tests disponible
Doit être installé avant: Aucun
Dépendances facultatives: *cpio*, *LLVM* (avec Clang) et *Rust-bindgen*

Linux API Headers

L'installation dépend de: Bash, Binutils, Coreutils, Findutils, GCC, Glibc, Grep, Gzip, Make, Perl et Sed
Requis à l'exécution: Aucun
La suite de tests dépend de: Aucune suite de tests disponible
Doit être installé avant: Aucun
Dépendances facultatives: Aucun

Lz4

L'installation dépend de: Bash, Binutils, Coreutils, GCC, Glibc et Make
Requis à l'exécution: Glibc
La suite de tests dépend de: Python
Doit être installé avant: Zstd
Dépendances facultatives: Aucun

M4

L'installation dépend de: Bash, Binutils, Coreutils, GCC, Glibc, Grep, Make, Sed, et Texinfo
Requis à l'exécution: Bash et Glibc
La suite de tests dépend de: Diffutils
Doit être installé avant: Autoconf et Bison
Dépendances facultatives: *libsigsegv*

Make

L'installation dépend de: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, Make, Sed et Texinfo
Requis à l'exécution: Glibc
La suite de tests dépend de: Perl et Procps-ng
Doit être installé avant: Aucun
Dépendances facultatives: *Guile*

Man-DB

L'installation dépend de: Bash, Binutils, Bzip2, Coreutils, Flex, GCC, GDBM, Gettext, Glibc, Grep, Groff, Gzip, Less, Libpipeline, Make, Pkgconf, Sed et Xz
Requis à l'exécution: Bash, GDBM, Groff, Glibc, Gzip, Less, Libpipeline et Zlib
La suite de tests dépend de: Util-linux
Doit être installé avant: Aucun
Dépendances facultatives: *libseccomp* et *po4a*

Man-Pages

L'installation dépend de: Bash, Coreutils, Make et Sed
Requis à l'exécution: Aucun
La suite de tests dépend de: Aucune suite de tests disponible
Doit être installé avant: Aucun
Dépendances facultatives: Aucun

MarkupSafe

L'installation dépend de: Python, Setuptools et Wheel
Requis à l'exécution: Python
La suite de tests dépend de: Aucune suite de tests disponible
Doit être installé avant: Jinja2
Dépendances facultatives: Aucun

Meson

L'installation dépend de: Ninja, Python, Setuptools et Wheel
Requis à l'exécution: Python
La suite de tests dépend de: Aucune suite de tests disponible
Doit être installé avant: Udev
Dépendances facultatives: Aucun

MPC

L'installation dépend de: Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, GMP, Make, MPFR, Sed et Texinfo

Requis à l'exécution: Glibc, GMP et MPFR

La suite de tests dépend de: Aucun

Doit être installé avant: GCC

Dépendances facultatives: Aucun

MPFR

L'installation dépend de: Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, GMP, Make, Sed et Texinfo

Requis à l'exécution: Glibc et GMP

La suite de tests dépend de: Aucun

Doit être installé avant: Gawk et GCC

Dépendances facultatives: Aucun

Ncurses

L'installation dépend de: Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Make, Patch et Sed

Requis à l'exécution: Glibc

La suite de tests dépend de: Aucune suite de tests disponible

Doit être installé avant: Bash, GRUB, Inetutils, Less, Procps-ng, Psmisc, Readline, Texinfo, Util-linux et Vim

Dépendances facultatives: Aucun

Ninja

L'installation dépend de: Binutils, Coreutils, GCC et Python

Requis à l'exécution: GCC et Glibc

La suite de tests dépend de: *cmake*

Doit être installé avant: Meson

Dépendances facultatives: *Asciidoc*, *Doxygen*, *Emacs* et *re2c*

OpenSSL

L'installation dépend de: Binutils, Coreutils, GCC, Make et Perl

Requis à l'exécution: Glibc et Perl

La suite de tests dépend de: Aucun

Doit être installé avant: Coreutils, Kmod, Linux et Udev

Dépendances facultatives: Aucun

Packaging

L'installation dépend de: Flit-core et Python
Requis à l'exécution: Python
La suite de tests dépend de: Aucune suite de tests disponible
Doit être installé avant: Wheel
Dépendances facultatives: *pytest*

Patch

L'installation dépend de: Attr, Bash, Binutils, Coreutils, GCC, Glibc, Grep, Make et Sed
Requis à l'exécution: Attr et Glibc
La suite de tests dépend de: Diffutils
Doit être installé avant: Aucun
Dépendances facultatives: *Ed*

Perl

L'installation dépend de: Bash, Binutils, Coreutils, Gawk, GCC, GDBM, Glibc, Grep, Libxcrypt, Make, Sed et Zlib
Requis à l'exécution: GDBM, Glibc et Libxcrypt
La suite de tests dépend de: Iana-Etc, Less et Procps-ng
Doit être installé avant: Autoconf
Dépendances facultatives: *Berkeley DB*

Pkgconf

L'installation dépend de: Bash, Binutils, Coreutils, Gawk, GCC, Glibc, Grep, Make et Sed
Requis à l'exécution: Glibc
La suite de tests dépend de: Aucun
Doit être installé avant: Binutils, E2fsprogs, IProute2, Kmod, Man-DB, Procps-ng, Python, Udev et Util-linux
Dépendances facultatives: Aucun

Procps-ng

L'installation dépend de: Bash, Binutils, Coreutils, GCC, Glibc, Make, Ncurses et Pkgconf
Requis à l'exécution: Glibc
La suite de tests dépend de: DejaGNU
Doit être installé avant: Aucun
Dépendances facultatives: *elogind*

Psmisc

L'installation dépend de: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, Make, Ncurses, et Sed
Requis à l'exécution: Glibc et Ncurses
La suite de tests dépend de: Expect
Doit être installé avant: Aucun
Dépendances facultatives: Aucun

Python

L'installation dépend de: Bash, Binutils, Coreutils, Expat, GCC, Gdbm, Gettext, Glibc, Grep, Libffi, Libxcrypt, Make, Ncurses, OpenSSL, Pkgconf, Sed et Util-linux
Requis à l'exécution: Bzip2, Expat, Gdbm, Glibc, Libffi, Libxcrypt, Ncurses, OpenSSL et Zlib
La suite de tests dépend de: GDB et Valgrind
Doit être installé avant: Ninja
Dépendances facultatives: *Berkeley DB*, *libnsl*, *SQLite* et *Tk*

Readline

L'installation dépend de: Bash, Binutils, Coreutils, Gawk, GCC, Glibc, Grep, Make, Ncurses, Patch, Sed et Texinfo
Requis à l'exécution: Glibc et Ncurses
La suite de tests dépend de: Aucune suite de tests disponible
Doit être installé avant: Bash, Bc et Gawk
Dépendances facultatives: Aucun

Sed

L'installation dépend de: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, Make, Sed et Texinfo
Requis à l'exécution: Acl, Attr et Glibc
La suite de tests dépend de: Diffutils et Gawk
Doit être installé avant: E2fsprogs, File, Libtool et Shadow
Dépendances facultatives: Aucun

Setuptools

L'installation dépend de: Python et Wheel
Requis à l'exécution: Python
La suite de tests dépend de: Aucune suite de tests disponible
Doit être installé avant: Jinja2, MarkupSafe et Meson
Dépendances facultatives: Aucun

Shadow

L'installation dépend de: Acl, Attr, Bash, Binutils, Coreutils, Diffutils, Findutils, Gawk, GCC, Gettext, Glibc, Grep, Libcap, Libxcrypt, Make et Sed

Requis à l'exécution: Glibc et Libxcrypt

La suite de tests dépend de: Aucune suite de tests disponible

Doit être installé avant: Coreutils

Dépendances facultatives: *CrackLib* et *Linux-PAM*

Sysklogd

L'installation dépend de: Binutils, Coreutils, GCC, Glibc, Make et Patch

Requis à l'exécution: Glibc

La suite de tests dépend de: Aucune suite de tests disponible

Doit être installé avant: Aucun

Dépendances facultatives: Aucun

SysVinit

L'installation dépend de: Binutils, Coreutils, GCC, Glibc, Make et Sed

Requis à l'exécution: Glibc

La suite de tests dépend de: Aucune suite de tests disponible

Doit être installé avant: Aucun

Dépendances facultatives: Aucun

Tar

L'installation dépend de: Acl, Attr, Bash, Binutils, Bison, Coreutils, GCC, Gettext, Glibc, Grep, Inetutils, Make, Sed et Texinfo

Requis à l'exécution: Acl, Attr, Bzip2, Glibc, Gzip et Xz

La suite de tests dépend de: Autoconf, Diffutils, Findutils, Gawk et Gzip

Doit être installé avant: Aucun

Dépendances facultatives: Aucun

Tcl

L'installation dépend de: Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Grep, Make et Sed

Requis à l'exécution: Glibc et Zlib

La suite de tests dépend de: Aucun

Doit être installé avant: Aucun

Dépendances facultatives: Aucun

Texinfo

L'installation dépend de: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, Make, Ncurses, Patch et Sed
Requis à l'exécution: Glibc et Ncurses
La suite de tests dépend de: Aucun
Doit être installé avant: Aucun
Dépendances facultatives: Aucun

Udev

L'installation dépend de: Acl, Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Gperf, Grep, Jinja2, Libcap, Libxcrypt, Meson, OpenSSL, Pkgconf, Sed, Util-linux et Zstd
Requis à l'exécution: Acl, Glibc, Libcap, OpenSSL et Util-linux
La suite de tests dépend de: Aucun
Doit être installé avant: Util-linux
Dépendances facultatives: Aucun

Util-linux

L'installation dépend de: Bash, Binutils, Coreutils, Diffutils, File, Findutils, Gawk, GCC, Gettext, Glibc, Grep, Make, Ncurses, Pkgconf, Sed, Udev et Zlib
Requis à l'exécution: Glibc, Ncurses, Readline, Udev et Zlib
La suite de tests dépend de: Aucun
Doit être installé avant: Aucun
Dépendances facultatives: *Asciidoctor*, *Libcap-NG*, *libeconf*, *libuser*, *libutempter*, *Linux-PAM*, *smartmontools*, *po4a* et *slang*

Vim

L'installation dépend de: Acl, Attr, Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Grep, Make, Ncurses et Sed
Requis à l'exécution: Acl, Attr, Glibc, Python, Ncurses et Tcl
La suite de tests dépend de: Aucun
Doit être installé avant: Aucun
Dépendances facultatives: *Xorg*, *GTK+2*, *LessTif*, *Ruby* et *GPM*

Wheel

L'installation dépend de: Python, Flit-core et packaging
Requis à l'exécution: Python
La suite de tests dépend de: Aucune suite de tests disponible
Doit être installé avant: Jinja2, MarkupSafe, Meson et Setuptools
Dépendances facultatives: Aucun

XML::Parser

L'installation dépend de: Bash, Binutils, Coreutils, Expat, GCC, Glibc, Make et Perl
Requis à l'exécution: Expat, Glibc et Perl
La suite de tests dépend de: Perl
Doit être installé avant: Intltool
Dépendances facultatives: *LWP::UserAgent*

Xz

L'installation dépend de: Bash, Binutils, Coreutils, Diffutils, GCC, Glibc et Make
Requis à l'exécution: Glibc
La suite de tests dépend de: Aucun
Doit être installé avant: File, GRUB, Kmod, Libelf, Man-DB et Udev
Dépendances facultatives: Aucun

Zlib

L'installation dépend de: Bash, Binutils, Coreutils, GCC, Glibc, Make et Sed
Requis à l'exécution: Glibc
La suite de tests dépend de: Aucun
Doit être installé avant: File, Kmod, Libelf, Perl et Util-linux
Dépendances facultatives: Aucun

Zstd

L'installation dépend de: Binutils, Coreutils, GCC, Glibc, Gzip, Lz4, Make, Xz et Zlib
Requis à l'exécution: Glibc
La suite de tests dépend de: Aucun
Doit être installé avant: Binutils, GCC, Libelf et Udev
Dépendances facultatives: Aucun

Annexe D. Scripts de démarrage et de sysconfig version-20250827

Les scripts présents dans cette annexe sont listés dans le répertoire où ils sont généralement stockés. L'ordre est le suivant : `/etc/rc.d/init.d`, `/etc/sysconfig`, `/etc/sysconfig/network-devices` et `/etc/sysconfig/network-devices/services`. Au sein de chaque section, les fichiers sont listés dans l'ordre où ils sont normalement appelés.

D.1. /etc/rc.d/init.d/rc

Le script `rc` est le premier script appelé par init : il initialise le processus de démarrage.

```
#!/bin/bash
#############################################################################
# Begin rc
#
# Description : Main Run Level Control Script
#
# Authors     : Gerard Beekmans  - gerard@linuxfromscratch.org
#             : DJ Lucas - dj@linuxfromscratch.org
# Updates     : Bruce Dubbs - bdubbs@linuxfromscratch.org
#             : Pierre Labastie - pierre@linuxfromscratch.org
#
# Version     : LFS 7.0
#
# Notes       : Updates March 24th, 2022: new semantics of S/K files
#               - Instead of testing that S scripts were K scripts in the
#                 previous runlevel, test that they were not S scripts
#               - Instead of testing that K scripts were S scripts in the
#                 previous runlevel, test that they were not K scripts
#               - S scripts in runlevel 0 or 6 are now run with
#                "script start" (was "script stop" previously).
#############################################################################

. /lib/lsb/init-functions

print_error_msg()
{
   log_failure_msg
   # $i is set when called
   MSG="FAILURE:\n\nYou should not be reading this error message.\n\n"
   MSG="${MSG}It means that an unforeseen error took place in\n"
   MSG="${MSG}${i},\n"
   MSG="${MSG}which exited with a return value of ${error_value}.\n"

   MSG="${MSG}If you're able to track this error down to a bug in one of\n"
   MSG="${MSG}the files provided by the ${DISTRO_MINI} book,\n"
   MSG="${MSG}please be so kind to inform us at ${DISTRO_CONTACT}.\n"
   log_failure_msg "${MSG}"

   log_info_msg "Press Enter to continue..."
   wait_for_user
}

check_script_status()
{
   # $i is set when called
   if [ ! -f ${i} ]; then
      log_warning_msg "${i} is not a valid symlink."
      SCRIPT_STAT="1"
   fi

   if [ ! -x ${i} ]; then
      log_warning_msg "${i} is not executable, skipping."
```

```
      SCRIPT_STAT="1"
   fi
}

run()
{
   if [ -z $interactive ]; then
      ${1} ${2}
      return $?
   fi

   while true; do
      read -p "Run ${1} ${2} (Yes/no/continue)? " -n 1 runit
      echo

      case ${runit} in
         c | C)
            interactive=""
            ${i} ${2}
            ret=${?}
            break;
            ;;

         n | N)
            return 0
            ;;

         y | Y)
            ${i} ${2}
            ret=${?}
            break
            ;;
      esac
   done

   return $ret
}

# Read any local settings/overrides
[ -r /etc/sysconfig/rc.site ] && source /etc/sysconfig/rc.site

DISTRO=${DISTRO:-"Linux From Scratch"}
DISTRO_CONTACT=${DISTRO_CONTACT:-"lfs-dev@lists.linuxfromscratch.org (Registration required)"}
DISTRO_MINI=${DISTRO_MINI:-"LFS"}
IPROMPT=${IPROMPT:-"no"}

# These 3 signals will not cause our script to exit
trap "" INT QUIT TSTP

[ "${1}" != "" ] && runlevel=${1}

if [ "${runlevel}" == "" ]; then
   echo "Usage: ${0} <runlevel>" >&2
   exit 1
fi

previous=${PREVLEVEL}
[ "${previous}" == "" ] && previous=N

if [ ! -d /etc/rc.d/rc${runlevel}.d ]; then
   log_info_msg "/etc/rc.d/rc${runlevel}.d does not exist.\n"
   exit 1
fi

if [ "$runlevel" == "6" -o "$runlevel" == "0" ]; then IPROMPT="no"; fi

# Note: In ${LOGLEVEL:-7}, it is ':' 'dash' '7', not minus 7
```

```
if [ "$runlevel" == "S" ]; then
   [ -r /etc/sysconfig/console ] && source /etc/sysconfig/console
   dmesg -n "${LOGLEVEL:-7}"
fi

if [ "${IPROMPT}" == "yes" -a "${runlevel}" == "S" ]; then
   # The total length of the distro welcome string, without escape codes
   wlen=${wlen:-$(echo "Welcome to ${DISTRO}" | wc -c )}
   welcome_message=${welcome_message:-"Welcome to ${INFO}${DISTRO}${NORMAL}"}

   # The total length of the interactive string, without escape codes
   ilen=${ilen:-$(echo "Press 'I' to enter interactive startup" | wc -c )}
   i_message=${i_message:-"Press '${FAILURE}I${NORMAL}' to enter interactive startup"}


   # dcol and icol are spaces before the message to center the message
   # on screen. itime is the amount of wait time for the user to press a key
   wcol=$(( ( ${COLUMNS} - ${wlen} ) / 2 ))
   icol=$(( ( ${COLUMNS} - ${ilen} ) / 2 ))
   itime=${itime:-"3"}

   echo -e "\n\n"
   echo -e "\\033[${wcol}G${welcome_message}"
   echo -e "\\033[${icol}G${i_message}${NORMAL}"
   echo ""
   read -t "${itime}" -n 1 interactive 2>&1 > /dev/null
fi

# Make lower case
[ "${interactive}" == "I" ] && interactive="i"
[ "${interactive}" != "i" ] && interactive=""

# Read the state file if it exists from runlevel S
[ -r /run/interactive ] && source /run/interactive

# Stop all services marked as K, except if marked as K in the previous
# runlevel: it is the responsibility of the script to not try to kill
# a non running service
if [ "${previous}" != "N" ]; then
   for i in $(ls -v /etc/rc.d/rc${runlevel}.d/K* 2> /dev/null)
   do
      check_script_status
      if [ "${SCRIPT_STAT}" == "1" ]; then
         SCRIPT_STAT="0"
         continue
      fi

      suffix=${i#/etc/rc.d/rc${runlevel}.d/K[0-9][0-9]}
      [ -e /etc/rc.d/rc${previous}.d/K[0-9][0-9]$suffix ] && continue

      run ${i} stop
      error_value=${?}

      if [ "${error_value}" != "0" ]; then print_error_msg; fi
   done
fi

if [ "${previous}" == "N" ]; then export IN_BOOT=1; fi

if [ "$runlevel" == "6" -a -n "${FASTBOOT}" ]; then
   touch /fastboot
fi


# Start all services marked as S in this runlevel, except if marked as
# S in the previous runlevel
# it is the responsibility of the script to not try to start an already running
```

```
# service
for i in $( ls -v /etc/rc.d/rc${runlevel}.d/S* 2> /dev/null)
do

   if [ "${previous}" != "N" ]; then
      suffix=${i#/etc/rc.d/rc${runlevel}.d/S[0-9][0-9]}
      [ -e /etc/rc.d/rc${previous}.d/S[0-9][0-9]$suffix ] && continue
   fi

   check_script_status
   if [ "${SCRIPT_STAT}" == "1" ]; then
      SCRIPT_STAT="0"
      continue
   fi

   run ${i} start

   error_value=${?}

   if [ "${error_value}" != "0" ]; then print_error_msg; fi
done

# Store interactive variable on switch from runlevel S and remove if not
if [ "${runlevel}" == "S" -a "${interactive}" == "i" ]; then
    echo "interactive=\"i\"" > /run/interactive
else
    rm -f /run/interactive 2> /dev/null
fi

# Copy the boot log on initial boot only
if [ "${previous}" == "N" -a  "${runlevel}" != "S" ]; then
   cat $BOOTLOG >> /var/log/boot.log

   # Mark the end of boot
   echo "--------" >> /var/log/boot.log

   # Remove the temporary file
   rm -f $BOOTLOG 2> /dev/null
fi

# End rc
```

D.2. /lib/lsb/init-functions

```
#!/bin/sh
########################################################################
#
# Begin /lib/lsb/init-funtions
#
# Description : Run Level Control Functions
#
# Authors     : Gerard Beekmans - gerard@linuxfromscratch.org
#             : DJ Lucas - dj@linuxfromscratch.org
# Update      : Bruce Dubbs - bdubbs@linuxfromscratch.org
#
# Version     : LFS 7.0
#
# Notes       : With code based on Matthias Benkmann's simpleinit-msb
#               http://winterdrache.de/linux/newboot/index.html
#
#               The file should be located in /lib/lsb
#
########################################################################

## Environmental setup
# Setup default values for environment
umask 022
```

```
export PATH="/bin:/usr/bin:/sbin:/usr/sbin"

## Set color commands, used via echo
# Please consult `man console_codes for more information
# under the "ECMA-48 Set Graphics Rendition" section
#
# Warning: when switching from a 8bit to a 9bit font,
# the linux console will reinterpret the bold (1;) to
# the top 256 glyphs of the 9bit font.  This does
# not affect framebuffer consoles

NORMAL="\\033[0;39m"         # Standard console grey
SUCCESS="\\033[1;32m"        # Success is green
WARNING="\\033[1;33m"        # Warnings are yellow
FAILURE="\\033[1;31m"        # Failures are red
INFO="\\033[1;36m"           # Information is light cyan
BRACKET="\\033[1;34m"        # Brackets are blue

# Use a colored prefix
BMPREFIX="      "
SUCCESS_PREFIX="${SUCCESS}  *  ${NORMAL} "
FAILURE_PREFIX="${FAILURE}*****${NORMAL} "
WARNING_PREFIX="${WARNING} *** ${NORMAL} "
SKIP_PREFIX="${INFO}  S   ${NORMAL}"

SUCCESS_SUFFIX="${BRACKET}[${SUCCESS}  OK  ${BRACKET}]${NORMAL}"
FAILURE_SUFFIX="${BRACKET}[${FAILURE} FAIL ${BRACKET}]${NORMAL}"
WARNING_SUFFIX="${BRACKET}[${WARNING} WARN ${BRACKET}]${NORMAL}"
SKIP_SUFFIX="${BRACKET}[${INFO} SKIP ${BRACKET}]${NORMAL}"

BOOTLOG=/run/bootlog
KILLDELAY=3
SCRIPT_STAT="0"

# Set any user specified environment variables e.g. HEADLESS
[ -r /etc/sysconfig/rc.site ]  && . /etc/sysconfig/rc.site

# If HEADLESS is set, use that.
# If file descriptor 1 or 2 (stdout and stderr) is not open or
# does not refer to a terminal, consider the script headless.
[ ! -t 1 -o ! -t 2 ] && HEADLESS=${HEADLESS:-yes}

if [ "x$HEADLESS" != "xyes" ]
then
  ## Screen Dimensions
  # Find current screen size
  if [ -z "${COLUMNS}" ]; then
    COLUMNS=$(stty size)
    COLUMNS=${COLUMNS##* }
  fi
else
   COLUMNS=80
fi

# When using remote connections, such as a serial port, stty size returns 0
if [ "${COLUMNS}" = "0" ]; then
   COLUMNS=80
fi

## Measurements for positioning result messages
COL=$((${COLUMNS} - 8))
WCOL=$((${COL} - 2))

## Set Cursor Position Commands, used via echo
SET_COL="\\033[${COL}G"      # at the $COL char
SET_WCOL="\\033[${WCOL}G"    # at the $WCOL char
CURS_UP="\\033[1A\\033[0G"   # Up one line, at the 0'th char
```

```
CURS_ZERO="\\033[0G"

################################################################################
# start_daemon()                                                               #
# Usage: start_daemon [-f] [-n nicelevel] [-p pidfile] pathname [args...]      #
#                                                                              #
# Purpose: This runs the specified program as a daemon                         #
#                                                                              #
# Inputs: -f: (force) run the program even if it is already running.           #
#         -n nicelevel: specify a nice level. See 'man nice(1)'.               #
#         -p pidfile: use the specified file to determine PIDs.                #
#         pathname: the complete path to the specified program                 #
#         args: additional arguments passed to the program (pathname)          #
#                                                                              #
# Return values (as defined by LSB exit codes):                                #
#       0 - program is running or service is OK                                #
#       1 - generic or unspecified error                                       #
#       2 - invalid or excessive argument(s)                                   #
#       5 - program is not installed                                           #
################################################################################
start_daemon()
{
    local force=""
    local nice="0"
    local pidfile=""
    local pidlist=""
    local retval=""

    # Process arguments
    while true
    do
        case "${1}" in

            -f)
                force="1"
                shift 1
                ;;

            -n)
                nice="${2}"
                shift 2
                ;;

            -p)
                pidfile="${2}"
                shift 2
                ;;

            -*)
                return 2
                ;;

            *)
                program="${1}"
                break
                ;;
        esac
    done

    # Check for a valid program
    if [ ! -e "${program}" ]; then return 5; fi

    # Execute
    if [ -z "${force}" ]; then
        if [ -z "${pidfile}" ]; then
            # Determine the pid by discovery
            pidlist=`pidofproc "${1}"`
```

```
            retval="${?}"
        else
            # The PID file contains the needed PIDs
            # Note that by LSB requirement, the path must be given to pidofproc,
            # however, it is not used by the current implementation or standard.
            pidlist=`pidofproc -p "${pidfile}" "${1}"`
            retval="${?}"
        fi

        # Return a value ONLY
        # It is the init script's (or distribution's functions) responsibility
        # to log messages!
        case "${retval}" in

            0)
                # Program is already running correctly, this is a
                # successful start.
                return 0
                ;;

            1)
                # Program is not running, but an invalid pid file exists
                # remove the pid file and continue
                rm -f "${pidfile}"
                ;;

            3)
                # Program is not running and no pidfile exists
                # do nothing here, let start_deamon continue.
                ;;

            *)
                # Others as returned by status values shall not be interpreted
                # and returned as an unspecified error.
                return 1
                ;;
        esac
    fi

    # Do the start!
    nice -n "${nice}" "${@}"
}

##############################################################################
# killproc()                                                                 #
# Usage: killproc [-p pidfile] pathname [signal]                             #
#                                                                            #
# Purpose: Send control signals to running processes                         #
#                                                                            #
# Inputs: -p pidfile, uses the specified pidfile                             #
#         pathname, pathname to the specified program                        #
#         signal, send this signal to pathname                               #
#                                                                            #
# Return values (as defined by LSB exit codes):                              #
#       0 - program (pathname) has stopped/is already stopped or a           #
#           running program has been sent specified signal and stopped       #
#           successfully                                                     #
#       1 - generic or unspecified error                                     #
#       2 - invalid or excessive argument(s)                                 #
#       5 - program is not installed                                         #
#       7 - program is not running and a signal was supplied                 #
##############################################################################
killproc()
{
    local pidfile
    local program
    local prefix
```

```
    local progname
    local signal="-TERM"
    local fallback="-KILL"
    local nosig
    local pidlist
    local retval
    local pid
    local delay="30"
    local piddead
    local dtime

    # Process arguments
    while true; do
        case "${1}" in
            -p)
                pidfile="${2}"
                shift 2
                ;;

             *)
                program="${1}"
                if [ -n "${2}" ]; then
                    signal="${2}"
                    fallback=""
                else
                    nosig=1
                fi

                # Error on additional arguments
                if [ -n "${3}" ]; then
                    return 2
                else
                    break
                fi
                ;;
        esac
    done

    # Check for a valid program
    if [ ! -e "${program}" ]; then return 5; fi

    # Check for a valid signal
    check_signal "${signal}"
    if [ "${?}" -ne "0" ]; then return 2; fi

    # Get a list of pids
    if [ -z "${pidfile}" ]; then
        # determine the pid by discovery
        pidlist=`pidofproc "${1}"`
        retval="${?}"
    else
        # The PID file contains the needed PIDs
        # Note that by LSB requirement, the path must be given to pidofproc,
        # however, it is not used by the current implementation or standard.
        pidlist=`pidofproc -p "${pidfile}" "${1}"`
        retval="${?}"
    fi

    # Return a value ONLY
    # It is the init script's (or distribution's functions) responsibility
    # to log messages!
    case "${retval}" in

        0)
            # Program is running correctly
            # Do nothing here, let killproc continue.
            ;;
```

```
    1)
        # Program is not running, but an invalid pid file exists
        # Remove the pid file.

        progname=${program##*/}

        if [[ -e "/run/${progname}.pid" ]]; then
            pidfile="/run/${progname}.pid"
            rm -f "${pidfile}"
        fi

        # This is only a success if no signal was passed.
        if [ -n "${nosig}" ]; then
            return 0
        else
            return 7
        fi
        ;;

    3)
        # Program is not running and no pidfile exists
        # This is only a success if no signal was passed.
        if [ -n "${nosig}" ]; then
            return 0
        else
            return 7
        fi
        ;;

    *)
        # Others as returned by status values shall not be interpreted
        # and returned as an unspecified error.
        return 1
        ;;
esac

# Perform different actions for exit signals and control signals
check_sig_type "${signal}"

if [ "${?}" -eq "0" ]; then # Signal is used to terminate the program

    # Account for empty pidlist (pid file still exists and no
    # signal was given)
    if [ "${pidlist}" != "" ]; then

        # Kill the list of pids
        for pid in ${pidlist}; do

            kill -0 "${pid}" 2> /dev/null

            if [ "${?}" -ne "0" ]; then
                # Process is dead, continue to next and assume all is well
                continue
            else
                kill "${signal}" "${pid}" 2> /dev/null

                # Wait up to ${delay}/10 seconds to for "${pid}" to
                # terminate in 10ths of a second

                while [ "${delay}" -ne "0" ]; do
                    kill -0 "${pid}" 2> /dev/null || piddead="1"
                    if [ "${piddead}" = "1" ]; then break; fi
                    sleep 0.1
                    delay="$(( ${delay} - 1 ))"
                done
```

```
                    # If a fallback is set, and program is still running, then
                    # use the fallback
                    if [ -n "${fallback}" -a "${piddead}" != "1" ]; then
                        kill "${fallback}" "${pid}" 2> /dev/null
                        sleep 1
                        # Check again, and fail if still running
                        kill -0 "${pid}" 2> /dev/null && return 1
                    fi
                fi
            done
        fi

        # Check for and remove stale PID files.
        if [ -z "${pidfile}" ]; then
            # Find the basename of $program
            prefix=`echo "${program}" | sed 's/[^/]*$//'`
            progname=`echo "${program}" | sed "s@${prefix}@@"`

            if [ -e "/run/${progname}.pid" ]; then
                rm -f "/run/${progname}.pid" 2> /dev/null
            fi
        else
            if [ -e "${pidfile}" ]; then rm -f "${pidfile}" 2> /dev/null; fi
        fi

    # For signals that do not expect a program to exit, simply
    # let kill do its job, and evaluate kill's return for value

    else # check_sig_type - signal is not used to terminate program
        for pid in ${pidlist}; do
            kill "${signal}" "${pid}"
            if [ "${?}" -ne "0" ]; then return 1; fi
        done
    fi
}

##############################################################################
# pidofproc()                                                                #
# Usage: pidofproc [-p pidfile] pathname                                     #
#                                                                            #
# Purpose: This function returns one or more pid(s) for a particular daemon  #
#                                                                            #
# Inputs: -p pidfile, use the specified pidfile instead of pidof             #
#         pathname, path to the specified program                            #
#                                                                            #
# Return values (as defined by LSB status codes):                            #
#       0 - Success (PIDs to stdout)                                         #
#       1 - Program is dead, PID file still exists (remaining PIDs output)   #
#       3 - Program is not running (no output)                               #
##############################################################################
pidofproc()
{
    local pidfile
    local program
    local prefix
    local progname
    local pidlist
    local lpids
    local exitstatus="0"

    # Process arguments
    while true; do
        case "${1}" in

            -p)
                pidfile="${2}"
                shift 2
```

```
                ;;

            *)
                program="${1}"
                if [ -n "${2}" ]; then
                    # Too many arguments
                    # Since this is status, return unknown
                    return 4
                else
                    break
                fi
                ;;
        esac
    done

    # If a PID file is not specified, try and find one.
    if [ -z "${pidfile}" ]; then
        # Get the program's basename
        prefix=`echo "${program}" | sed 's/[^/]*$//'`

        if [ -z "${prefix}" ]; then
           progname="${program}"
        else
           progname=`echo "${program}" | sed "s@${prefix}@@"`
        fi

        # If a PID file exists with that name, assume that is it.
        if [ -e "/run/${progname}.pid" ]; then
            pidfile="/run/${progname}.pid"
        fi
    fi

    # If a PID file is set and exists, use it.
    if [ -n "${pidfile}" -a -e "${pidfile}" ]; then
        # Use the value in the first line of the pidfile
        pidlist=`/bin/head -n1 "${pidfile}"`
    else
        # Use pidof
        pidlist=`pidof "${program}"`
    fi

    # Figure out if all listed PIDs are running.
    for pid in ${pidlist}; do
        kill -0 ${pid} 2> /dev/null

        if [ "${?}" -eq "0" ]; then
            lpids="${lpids}${pid} "
        else
            exitstatus="1"
        fi
    done

    if [ -z "${lpids}" -a ! -f "${pidfile}" ]; then
        return 3
    else
        echo "${lpids}"
        return "${exitstatus}"
    fi
}

########################################################################################
# statusproc()                                                                         #
# Usage: statusproc [-p pidfile] pathname                                              #
#                                                                                      #
# Purpose: This function prints the status of a particular daemon to stdout            #
#                                                                                      #
# Inputs: -p pidfile, use the specified pidfile instead of pidof                       #
```

```
#         pathname, path to the specified program                              #
#                                                                              #
# Return values:                                                               #
#       0 - Status printed                                                     #
#       1 - Input error. The daemon to check was not specified.                #
################################################################################
statusproc()
{
   local pidfile
   local pidlist

   if [ "${#}" = "0" ]; then
      echo "Usage: statusproc [-p pidfle] {program}"
      exit 1
   fi

   # Process arguments
   while true; do
       case "${1}" in

           -p)
               pidfile="${2}"
               shift 2
               ;;

           *)
               if [ -n "${2}" ]; then
                   echo "Too many arguments"
                   return 1
               else
                   break
               fi
               ;;
       esac
   done

   if [ -n "${pidfile}" ]; then
      pidlist=`pidofproc -p "${pidfile}" $@`
   else
      pidlist=`pidofproc $@`
   fi

   # Trim trailing blanks
   pidlist=`echo "${pidlist}" | sed -r 's/ +$//'`

   base="${1##*/}"

   if [ -n "${pidlist}" ]; then
      /bin/echo -e "${INFO}${base} is running with Process" \
         "ID(s) ${pidlist}.${NORMAL}"
   else
      if [ -n "${base}" -a -e "/run/${base}.pid" ]; then
         /bin/echo -e "${WARNING}${1} is not running but" \
            "/run/${base}.pid exists.${NORMAL}"
      else
         if [ -n "${pidfile}" -a -e "${pidfile}" ]; then
            /bin/echo -e "${WARNING}${1} is not running" \
               "but ${pidfile} exists.${NORMAL}"
         else
            /bin/echo -e "${INFO}${1} is not running.${NORMAL}"
         fi
      fi
   fi
}

################################################################################
# timespec()                                                                   #
```

```
#                                                                              #
# Purpose: An internal utility function to format a timestamp                  #
#          a boot log file.  Sets the STAMP variable.                          #
#                                                                              #
# Return value: Not used                                                       #
################################################################################
timespec()
{
   STAMP="$(echo `date +"%b %d %T %:z"` `hostname`) "
   return 0
}

################################################################################
# log_success_msg()                                                            #
# Usage: log_success_msg ["message"]                                           #
#                                                                              #
# Purpose: Print a successful status message to the screen and                 #
#          a boot log file.                                                    #
#                                                                              #
# Inputs: $@ - Message                                                         #
#                                                                              #
# Return values: Not used                                                      #
################################################################################
log_success_msg()
{
    if [ "x$HEADLESS" != "xyes" ]
    then
      /bin/echo -n -e "${BMPREFIX}${@}"
      /bin/echo -e "${CURS_ZERO}${SUCCESS_PREFIX}${SET_COL}${SUCCESS_SUFFIX}"
    else
      logmessage=`echo "${@}" | sed 's/\\\033[^a-zA-Z]*.//g'`
      /bin/echo -e "${logmessage} OK"
    fi
    # Strip non-printable characters from log file
    logmessage=`echo "${@}" | sed 's/\\\033[^a-zA-Z]*.//g'`

    timespec
    /bin/echo -e "${STAMP} ${logmessage} OK" >> ${BOOTLOG}

    return 0
}

log_success_msg2()
{
    if [ "x$HEADLESS" != "xyes" ]
    then
      /bin/echo -n -e "${BMPREFIX}${@}"
      /bin/echo -e "${CURS_ZERO}${SUCCESS_PREFIX}${SET_COL}${SUCCESS_SUFFIX}"
    else
      echo " OK"
    fi

    echo " OK" >> ${BOOTLOG}

    return 0
}

################################################################################
# log_failure_msg()                                                            #
# Usage: log_failure_msg ["message"]                                           #
#                                                                              #
# Purpose: Print a failure status message to the screen and                    #
#          a boot log file.                                                    #
#                                                                              #
# Inputs: $@ - Message                                                         #
#                                                                              #
# Return values: Not used                                                      #
```

```
########################################################################################
log_failure_msg()
{
    if [ "x$HEADLESS" != "xyes" ]
    then
      /bin/echo -n -e "${BMPREFIX}${@}"
      /bin/echo -e "${CURS_ZERO}${FAILURE_PREFIX}${SET_COL}${FAILURE_SUFFIX}"
    else
      logmessage=`echo "${@}" | sed 's/\\\033[^a-zA-Z]*.//g'`
      /bin/echo -e "${logmessage} FAIL"
    fi

    # Strip non-printable characters from log file

    timespec
    logmessage=`echo "${@}" | sed 's/\\\033[^a-zA-Z]*.//g'`
    /bin/echo -e "${STAMP} ${logmessage} FAIL" >> ${BOOTLOG}

    return 0
}

log_failure_msg2()
{
    if [ "x$HEADLESS" != "xyes" ]
    then
      /bin/echo -n -e "${BMPREFIX}${@}"
      /bin/echo -e "${CURS_ZERO}${FAILURE_PREFIX}${SET_COL}${FAILURE_SUFFIX}"
    else
      echo "FAIL"
    fi

    echo "FAIL" >> ${BOOTLOG}

    return 0
}

########################################################################################
# log_warning_msg()                                                                    #
# Usage: log_warning_msg ["message"]                                                   #
#                                                                                      #
# Purpose: Print a warning status message to the screen and                            #
#          a boot log file.                                                            #
#                                                                                      #
# Return values: Not used                                                              #
########################################################################################
log_warning_msg()
{
    if [ "x$HEADLESS" != "xyes" ]
    then
      /bin/echo -n -e "${BMPREFIX}${@}"
      /bin/echo -e "${CURS_ZERO}${WARNING_PREFIX}${SET_COL}${WARNING_SUFFIX}"
    else
      logmessage=`echo "${@}" | sed 's/\\\033[^a-zA-Z]*.//g'`
      /bin/echo -e "${logmessage} WARN"
    fi

    # Strip non-printable characters from log file
    logmessage=`echo "${@}" | sed 's/\\\033[^a-zA-Z]*.//g'`
    timespec
    /bin/echo -e "${STAMP} ${logmessage} WARN" >> ${BOOTLOG}

    return 0
}

log_skip_msg()
{
    if [ "x$HEADLESS" != "xyes" ]
```

```
    then
      /bin/echo -n -e "${BMPREFIX}${@}"
      /bin/echo -e "${CURS_ZERO}${SKIP_PREFIX}${SET_COL}${SKIP_SUFFIX}"
    else
      logmessage=`echo "${@}" | sed 's/\\\033[^a-zA-Z]*.//g'`
      /bin/echo "SKIP"
    fi

    # Strip non-printable characters from log file
    logmessage=`echo "${@}" | sed 's/\\\033[^a-zA-Z]*.//g'`
    /bin/echo "SKIP" >> ${BOOTLOG}

    return 0
}

########################################################################################
# log_info_msg()                                                                       #
# Usage: log_info_msg message                                                          #
#                                                                                      #
# Purpose: Print an information message to the screen and                              #
#          a boot log file.  Does not print a trailing newline character.              #
#                                                                                      #
# Return values: Not used                                                              #
########################################################################################
log_info_msg()
{
    if [ "x$HEADLESS" != "xyes" ]
    then
      /bin/echo -n -e "${BMPREFIX}${@}"
    else
      logmessage=`echo "${@}" | sed 's/\\\033[^a-zA-Z]*.//g'`
      /bin/echo -n -e "${logmessage}"
    fi

    # Strip non-printable characters from log file
    logmessage=`echo "${@}" | sed 's/\\\033[^a-zA-Z]*.//g'`
    timespec
    /bin/echo -n -e "${STAMP} ${logmessage}" >> ${BOOTLOG}

    return 0
}

log_info_msg2()
{
    if [ "x$HEADLESS" != "xyes" ]
    then
      /bin/echo -n -e "${@}"
    else
      logmessage=`echo "${@}" | sed 's/\\\033[^a-zA-Z]*.//g'`
      /bin/echo -n -e "${logmessage}"
    fi

    # Strip non-printable characters from log file
    logmessage=`echo "${@}" | sed 's/\\\033[^a-zA-Z]*.//g'`
    /bin/echo -n -e "${logmessage}" >> ${BOOTLOG}

    return 0
}

########################################################################################
# evaluate_retval()                                                                    #
# Usage: Evaluate a return value and print success or failure as appropriate  #
#                                                                                      #
# Purpose: Convenience function to terminate an info message                           #
#                                                                                      #
# Return values: Not used                                                              #
########################################################################################
```

```
evaluate_retval()
{
   local error_value="${?}"

   if [ ${error_value} = 0 ]; then
      log_success_msg2
   else
      log_failure_msg2
   fi
}

########################################################################################
# check_signal()                                                                       #
# Usage: check_signal [ -{signal} ]                                                    #
#                                                                                      #
# Purpose: Check for a valid signal.  This is not defined by any LSB draft,    #
#          however, it is required to check the signals to determine if the    #
#          signals chosen are invalid arguments to the other functions.        #
#                                                                                      #
# Inputs: Accepts a single string value in the form of -{signal}               #
#                                                                                      #
# Return values:                                                                       #
#       0 - Success (signal is valid                                                   #
#       1 - Signal is not valid                                                        #
########################################################################################
check_signal()
{
    local valsig

    # Add error handling for invalid signals
    valsig=" -ALRM -HUP -INT -KILL -PIPE -POLL -PROF -TERM -USR1 -USR2"
    valsig="${valsig} -VTALRM -STKFLT -PWR -WINCH -CHLD -URG -TSTP -TTIN"
    valsig="${valsig} -TTOU -STOP -CONT -ABRT -FPE -ILL -QUIT -SEGV -TRAP"
    valsig="${valsig} -SYS -EMT -BUS -XCPU -XFSZ -0 -1 -2 -3 -4 -5 -6 -8 -9"
    valsig="${valsig} -11 -13 -14 -15 "

    echo "${valsig}" | grep -- " ${1} " > /dev/null

    if [ "${?}" -eq "0" ]; then
        return 0
    else
        return 1
    fi
}

########################################################################################
# check_sig_type()                                                                     #
# Usage: check_signal [ -{signal} | {signal} ]                                         #
#                                                                                      #
# Purpose: Check if signal is a program termination signal or a control signal #
#          This is not defined by any LSB draft, however, it is required to    #
#          check the signals to determine if they are intended to end a        #
#          program or simply to control it.                                    #
#                                                                                      #
# Inputs: Accepts a single string value in the form or -{signal} or {signal}   #
#                                                                                      #
# Return values:                                                                       #
#       0 - Signal is used for program termination                                     #
#       1 - Signal is used for program control                                         #
########################################################################################
check_sig_type()
{
    local valsig

    # The list of termination signals (limited to generally used items)
    valsig=" -ALRM -INT -KILL -TERM -PWR -STOP -ABRT -QUIT -2 -3 -6 -9 -14 -15 "
```

```
    echo "${valsig}" | grep -- " ${1} " > /dev/null

    if [ "${?}" -eq "0" ]; then
        return 0
    else
        return 1
    fi
}

########################################################################################
# wait_for_user()                                                                      #
#                                                                                      #
# Purpose: Wait for the user to respond if not a headless system                       #
#                                                                                      #
########################################################################################
wait_for_user()
{
   # Wait for the user by default
   [ "${HEADLESS=0}" = "0" ] && read ENTER
   return 0
}

########################################################################################
# is_true()                                                                            #
#                                                                                      #
# Purpose: Utility to test if a variable is true | yes | 1                             #
#                                                                                      #
########################################################################################
is_true()
{
   [ "$1" = "1" ] || [ "$1" = "yes" ] || [ "$1" = "true" ] ||  [ "$1" = "y" ] ||
   [ "$1" = "t" ]
}

# End /lib/lsb/init-functions
```

D.3. /etc/rc.d/init.d/mountvirtfs

```
#!/bin/sh
########################################################################
# Begin mountvirtfs
#
# Description : Ensure proc, sysfs, run, and dev are mounted
#
# Authors     : Gerard Beekmans - gerard@linuxfromscratch.org
#               DJ Lucas - dj@linuxfromscratch.org
# Update      : Bruce Dubbs - bdubbs@linuxfromscratch.org
#               Xi Ruoyao - xry111@xry111.site
#
# Version     : LFS 12.0
#
########################################################################

### BEGIN INIT INFO
# Provides:            mountvirtfs
# Required-Start:      $first
# Should-Start:
# Required-Stop:
# Should-Stop:
# Default-Start:       S
# Default-Stop:
# Short-Description:   Mounts various special fs needed at start
# Description:         Mounts /sys and /proc virtual (kernel) filesystems.
#                      Mounts /run (tmpfs) and /dev (devtmpfs).
#                      This is done only if they are not already mounted.
#                      with the kernel config proposed in the book, dev
#                      should be automatically mounted by the kernel.
```

```
# X-LFS-Provided-By:   LFS
### END INIT INFO

. /lib/lsb/init-functions

case "${1}" in
   start)
      # Make sure /run is available before logging any messages
      if ! mountpoint /run >/dev/null; then
         mount /run || failed=1
      fi

      mkdir -p   /run/lock
      chmod 1777 /run/lock

      log_info_msg "Mounting virtual file systems: ${INFO}/run"

      if ! mountpoint /proc >/dev/null; then
         log_info_msg2 " ${INFO}/proc"
         mount -o nosuid,noexec,nodev /proc || failed=1
      fi

      if ! mountpoint /sys >/dev/null; then
         log_info_msg2 " ${INFO}/sys"
         mount -o nosuid,noexec,nodev /sys || failed=1
      fi

      if ! mountpoint /dev >/dev/null; then
         log_info_msg2 " ${INFO}/dev"
         mount -o mode=0755,nosuid /dev  || failed=1
      fi

      mkdir -p /dev/shm
      log_info_msg2 " ${INFO}/dev/shm"
      mount -o nosuid,nodev /dev/shm || failed=1

      mkdir -p /sys/fs/cgroup
      log_info_msg2 " ${INFO}/sys/fs/cgroup"
      mount -o nosuid,noexec,nodev /sys/fs/cgroup || failed=1

      (exit ${failed})
      evaluate_retval
      if [ "${failed}" = 1 ]; then
         exit 1
      fi

      log_info_msg "Create symlinks in /dev targeting /proc: ${INFO}/dev/stdin"
      ln -sf /proc/self/fd/0 /dev/stdin  || failed=1

      log_info_msg2 " ${INFO}/dev/stdout"
      ln -sf /proc/self/fd/1 /dev/stdout || failed=1

      log_info_msg2 " ${INFO}/dev/stderr"
      ln -sf /proc/self/fd/2 /dev/stderr || failed=1

      log_info_msg2 " ${INFO}/dev/fd"
      ln -sfn /proc/self/fd  /dev/fd     || failed=1

      if [ -e /proc/kcore ]; then
         log_info_msg2 " ${INFO}/dev/core"
         ln -sf /proc/kcore  /dev/core   || failed=1
      fi

      (exit ${failed})
      evaluate_retval
      exit $failed
      ;;
```

```
   *)
      echo "Usage: ${0} {start}"
      exit 1
      ;;
esac

# End mountvirtfs
```

D.4. /etc/rc.d/init.d/modules

```
#!/bin/sh
########################################################################
# Begin modules
#
# Description : Module auto-loading script
#
# Authors     : Zack Winkles
#               DJ Lucas - dj@linuxfromscratch.org
# Update      : Bruce Dubbs - bdubbs@linuxfromscratch.org
#
# Version     : LFS 7.0
#
########################################################################

### BEGIN INIT INFO
# Provides:            modules
# Required-Start:      mountvirtfs
# Should-Start:
# Required-Stop:
# Should-Stop:
# Default-Start:       S
# Default-Stop:
# Short-Description:   Loads required modules.
# Description:         Loads modules listed in /etc/sysconfig/modules.
# X-LFS-Provided-By:   LFS
### END INIT INFO

# Assure that the kernel has module support.
[ -e /proc/modules ] || exit 0

. /lib/lsb/init-functions

case "${1}" in
   start)
      # Exit if there's no modules file or there are no
      # valid entries
      [ -r /etc/sysconfig/modules ]             || exit 0
      grep -E -qv '^($|#)' /etc/sysconfig/modules || exit 0

      log_info_msg "Loading modules:"

      # Only try to load modules if the user has actually given us
      # some modules to load.

      while read module args; do

         # Ignore comments and blank lines.
         case "$module" in
            ""|"#"*) continue ;;
         esac

         # Attempt to load the module, passing any arguments provided.
         modprobe ${module} ${args} >/dev/null

         # Print the module name if successful, otherwise take note.
         if [ $? -eq 0 ]; then
```

```
            log_info_msg2 " ${module}"
         else
            failedmod="${failedmod} ${module}"
         fi
      done < /etc/sysconfig/modules

      # Print a message about successfully loaded modules on the correct line.
      log_success_msg2

      # Print a failure message with a list of any modules that
      # may have failed to load.
      if [ -n "${failedmod}" ]; then
         log_failure_msg "Failed to load modules:${failedmod}"
         exit 1
      fi
      ;;

   *)
      echo "Usage: ${0} {start}"
      exit 1
      ;;
esac

exit 0

# End modules
```

D.5. /etc/rc.d/init.d/udev

```
#!/bin/sh
########################################################################
# Begin udev
#
# Description : Udev cold-plugging script
#
# Authors     : Zack Winkles, Alexander E. Patrakov
#               DJ Lucas - dj@linuxfromscratch.org
# Update      : Bruce Dubbs - bdubbs@linuxfromscratch.org
#               Xi Ruoyao - xry111@xry111.site
#
# Version     : LFS 12.0
#
########################################################################

### BEGIN INIT INFO
# Provides:            udev $time
# Required-Start:      localnet
# Should-Start:        modules
# Required-Stop:
# Should-Stop:
# Default-Start:       S
# Default-Stop:
# Short-Description:   Populates /dev with device nodes.
# Description:         Mounts a tempfs on /dev and starts the udevd daemon.
#                      Device nodes are created as defined by udev.
# X-LFS-Provided-By:   LFS
### END INIT INFO

. /lib/lsb/init-functions

case "${1}" in
   start)
      log_info_msg "Populating /dev with device nodes... "
      if ! grep -q '[[:space:]]sysfs' /proc/mounts; then
         log_failure_msg2
         msg="FAILURE:\n\nUnable to create "
         msg="${msg}devices without a SysFS filesystem\n\n"
```

```
         msg="${msg}After you press Enter, this system "
         msg="${msg}will be halted and powered off.\n\n"
         log_info_msg "$msg"
         log_info_msg "Press Enter to continue..."
         wait_for_user
         /etc/rc.d/init.d/halt start
      fi

      # Start the udev daemon to continually watch for, and act on,
      # uevents
      SYSTEMD_LOG_TARGET=kmsg /sbin/udevd --daemon

      # Now traverse /sys in order to "coldplug" devices that have
      # already been discovered
      /bin/udevadm trigger --action=add    --type=subsystems
      /bin/udevadm trigger --action=add    --type=devices
      /bin/udevadm trigger --action=change --type=devices

      # Now wait for udevd to process the uevents we triggered
      if ! is_true "$OMIT_UDEV_SETTLE"; then
         /bin/udevadm settle
      fi

      # If any LVM based partitions are on the system, ensure they
      # are activated so they can be used.
      if [ -x /sbin/vgchange ]; then /sbin/vgchange -a y >/dev/null; fi

      log_success_msg2
      ;;

   *)
      echo "Usage ${0} {start}"
      exit 1
      ;;
esac

exit 0

# End udev
```

D.6. /etc/rc.d/init.d/swap

```
#!/bin/sh
########################################################################
# Begin swap
#
# Description : Swap Control Script
#
# Authors     : Gerard Beekmans - gerard@linuxfromscratch.org
#               DJ Lucas - dj@linuxfromscratch.org
# Update      : Bruce Dubbs - bdubbs@linuxfromscratch.org
#
# Version     : LFS 7.0
#
########################################################################

### BEGIN INIT INFO
# Provides:            swap
# Required-Start:      udev
# Should-Start:        modules
# Required-Stop:       localnet
# Should-Stop:         $local_fs
# Default-Start:       S
# Default-Stop:        0 6
# Short-Description:   Activates and deactivates swap partitions.
# Description:         Activates and deactivates swap partitions defined in
#                      /etc/fstab.
```

```
# X-LFS-Provided-By:   LFS
### END INIT INFO

. /lib/lsb/init-functions

case "${1}" in
   start)
      log_info_msg "Activating all swap files/partitions..."
      swapon -a
      evaluate_retval
      ;;

   stop)
      log_info_msg "Deactivating all swap files/partitions..."
      swapoff -a
      evaluate_retval
      ;;

   restart)
      ${0} stop
      sleep 1
      ${0} start
      ;;

   status)
      log_success_msg "Retrieving swap status."
      swapon -s
      ;;

   *)
      echo "Usage: ${0} {start|stop|restart|status}"
      exit 1
      ;;
esac

exit 0

# End swap
```

D.7. /etc/rc.d/init.d/setclock

```
#!/bin/sh
########################################################################
# Begin setclock
#
# Description : Setting Linux Clock
#
# Authors     : Gerard Beekmans - gerard@linuxfromscratch.org
#               DJ Lucas - dj@linuxfromscratch.org
# Update      : Bruce Dubbs - bdubbs@linuxfromscratch.org
#
# Version     : LFS 7.0
#
########################################################################

### BEGIN INIT INFO
# Provides:
# Required-Start:
# Should-Start:        modules
# Required-Stop:
# Should-Stop:         $syslog
# Default-Start:       S
# Default-Stop:
# Short-Description:   Stores and restores time from the hardware clock
# Description:         On boot, system time is obtained from hwclock.  The
#                      hardware clock can also be set on shutdown.
# X-LFS-Provided-By:   LFS
```

```
### END INIT INFO

. /lib/lsb/init-functions

[ -r /etc/sysconfig/clock ] && . /etc/sysconfig/clock


case "${UTC}" in
   yes|true|1)
      CLOCKPARAMS="${CLOCKPARAMS} --utc"
      ;;

   no|false|0)
      CLOCKPARAMS="${CLOCKPARAMS} --localtime"
      ;;

esac

case ${1} in
   start)
      hwclock --hctosys ${CLOCKPARAMS} >/dev/null
      ;;

   stop)
      log_info_msg "Setting hardware clock..."
      hwclock --systohc ${CLOCKPARAMS} >/dev/null
      evaluate_retval
      ;;

   *)
      echo "Usage: ${0} {start|stop}"
      exit 1
      ;;

esac

exit 0
```

D.8. /etc/rc.d/init.d/checkfs

```
#!/bin/sh
########################################################################
# Begin checkfs
#
# Description : File System Check
#
# Authors     : Gerard Beekmans - gerard@linuxfromscratch.org
#               A. Luebke - luebke@users.sourceforge.net
#               DJ Lucas - dj@linuxfromscratch.org
# Update      : Bruce Dubbs - bdubbs@linuxfromscratch.org
#
# Version     : LFS 7.0
#
# Based on checkfs script from LFS-3.1 and earlier.
#
# From man fsck
# 0    - No errors
# 1    - File system errors corrected
# 2    - System should be rebooted
# 4    - File system errors left uncorrected
# 8    - Operational error
# 16   - Usage or syntax error
# 32   - Fsck canceled by user request
# 128  - Shared library error
#
########################################################################
```

```
### BEGIN INIT INFO
# Provides:            checkfs
# Required-Start:      udev swap
# Should-Start:
# Required-Stop:
# Should-Stop:
# Default-Start:       S
# Default-Stop:
# Short-Description:   Checks local filesystems before mounting.
# Description:         Checks local filesystems before mounting.
# X-LFS-Provided-By:   LFS
### END INIT INFO

. /lib/lsb/init-functions

case "${1}" in
   start)
      if [ -f /fastboot ]; then
         msg="/fastboot found, will omit "
         msg="${msg} file system checks as requested.\n"
         log_info_msg "${msg}"
         exit 0
      fi

      log_info_msg "Mounting root file system in read-only mode... "
      mount -n -o remount,ro / >/dev/null

      if [ ${?} != 0 ]; then
         log_failure_msg2
         msg="\n\nCannot check root "
         msg="${msg}filesystem because it could not be mounted "
         msg="${msg}in read-only mode.\n\n"
         msg="${msg}After you press Enter, this system will be "
         msg="${msg}halted and powered off.\n\n"
         log_failure_msg "${msg}"

         log_info_msg "Press Enter to continue..."
         wait_for_user
         /etc/rc.d/init.d/halt start
      else
         log_success_msg2
      fi

      if [ -f /forcefsck ]; then
         msg="/forcefsck found, forcing file"
         msg="${msg} system checks as requested."
         log_success_msg "$msg"
         options="-f"
      else
         options=""
      fi

      log_info_msg "Checking file systems..."
      # Note: -a option used to be -p; but this fails e.g. on fsck.minix
      if is_true "$VERBOSE_FSCK"; then
        fsck ${options} -a -A -C -T
      else
        fsck ${options} -a -A -C -T >/dev/null
      fi

      error_value=${?}

      if [ "${error_value}" = 0 ]; then
         log_success_msg2
      fi

      if [ "${error_value}" = 1 ]; then
```

```
         msg="\nWARNING:\n\nFile system errors "
         msg="${msg}were found and have been corrected.\n"
         msg="${msg}      You may want to double-check that "
         msg="${msg}everything was fixed properly."
         log_warning_msg "$msg"
      fi

      if [ "${error_value}" = 2 -o "${error_value}" = 3 ]; then
         msg="\nWARNING:\n\nFile system errors "
         msg="${msg}were found and have been "
         msg="${msg}corrected, but the nature of the "
         msg="${msg}errors require this system to be rebooted.\n\n"
         msg="${msg}After you press enter, "
         msg="${msg}this system will be rebooted\n\n"
         log_failure_msg "$msg"

         log_info_msg "Press Enter to continue..."
         wait_for_user
         reboot -f
      fi

      if [ "${error_value}" -gt 3 -a "${error_value}" -lt 16 ]; then
         msg="\nFAILURE:\n\nFile system errors "
         msg="${msg}were encountered that could not be "
         msg="${msg}fixed automatically.\nThis system "
         msg="${msg}cannot continue to boot and will "
         msg="${msg}therefore be halted until those "
         msg="${msg}errors are fixed manually by a "
         msg="${msg}System Administrator.\n\n"
         msg="${msg}After you press Enter, this system will be "
         msg="${msg}halted and powered off.\n\n"
         log_failure_msg "$msg"

         log_info_msg "Press Enter to continue..."
         wait_for_user
         /etc/rc.d/init.d/halt start
      fi

      if [ "${error_value}" -ge 16 ]; then
         msg="FAILURE:\n\nUnexpected failure "
         msg="${msg}running fsck.  Exited with error "
         msg="${msg} code: ${error_value}.\n"
         log_info_msg $msg
         exit ${error_value}
      fi

      exit 0
      ;;
   *)
      echo "Usage: ${0} {start}"
      exit 1
      ;;
esac

# End checkfs
```

D.9. /etc/rc.d/init.d/mountfs

```
#!/bin/sh
########################################################################
# Begin mountfs
#
# Description : File System Mount Script
#
# Authors     : Gerard Beekmans - gerard@linuxfromscratch.org
#               DJ Lucas - dj@linuxfromscratch.org
# Update      : Bruce Dubbs - bdubbs@linuxfromscratch.org
```

```
#
# Version     : LFS 7.0
#
###############################################################################

### BEGIN INIT INFO
# Provides:            $local_fs
# Required-Start:      udev checkfs
# Should-Start:        modules
# Required-Stop:       localnet
# Should-Stop:
# Default-Start:       S
# Default-Stop:        0 6
# Short-Description:   Mounts/unmounts local filesystems defined in /etc/fstab.
# Description:         Remounts root filesystem read/write and mounts all
#                      remaining local filesystems defined in /etc/fstab on
#                      start.  Remounts root filesystem read-only and unmounts
#                      remaining filesystems on stop.
# X-LFS-Provided-By:   LFS
### END INIT INFO

. /lib/lsb/init-functions

case "${1}" in
   start)
      log_info_msg "Remounting root file system in read-write mode..."
      mount --options remount,rw / >/dev/null
      evaluate_retval

      # Remove fsck-related file system watermarks.
      rm -f /fastboot /forcefsck

      # Make sure /dev/pts exists
      mkdir -p /dev/pts

      # This will mount all filesystems that do not have _netdev in
      # their option list.  _netdev denotes a network filesystem.

      log_info_msg "Mounting remaining file systems..."
      failed=0
      mount --all --test-opts no_netdev >/dev/null || failed=1
      evaluate_retval
      exit $failed
      ;;

   stop)
      # Don't unmount virtual file systems like /run
      log_info_msg "Unmounting all other currently mounted file systems..."
      # Ensure any loop devices are removed
      losetup -D
      umount --all --detach-loop --read-only \
             --types notmpfs,nosysfs,nodevtmpfs,noproc,nodevpts >/dev/null
      evaluate_retval

      # Make sure / is mounted read only (umount bug)
      mount --options remount,ro /

      # Make all LVM volume groups unavailable, if appropriate
      # This fails if swap or / are on an LVM partition
      #if [ -x /sbin/vgchange ]; then /sbin/vgchange -an > /dev/null; fi
      if [ -r /etc/mdadm.conf ]; then
         log_info_msg "Mark arrays as clean..."
         mdadm --wait-clean --scan
         evaluate_retval
      fi
      ;;
```

```
   *)
      echo "Usage: ${0} {start|stop}"
      exit 1
      ;;
esac

# End mountfs
```

D.10. /etc/rc.d/init.d/udev_retry

```
#!/bin/sh
########################################################################
# Begin udev_retry
#
# Description : Udev cold-plugging script (retry)
#
# Authors     : Alexander E. Patrakov
#               DJ Lucas - dj@linuxfromscratch.org
# Update      : Bruce Dubbs - bdubbs@linuxfromscratch.org
#               Bryan Kadzban -
#
# Version     : LFS 7.0
#
########################################################################

### BEGIN INIT INFO
# Provides:            udev_retry
# Required-Start:      udev
# Should-Start:        $local_fs cleanfs
# Required-Stop:
# Should-Stop:
# Default-Start:       S
# Default-Stop:
# Short-Description:   Replays failed uevents and creates additional devices.
# Description:         Replays any failed uevents that were skipped due to
#                      slow hardware initialization, and creates those needed
#                      device nodes
# X-LFS-Provided-By:   LFS
### END INIT INFO

. /lib/lsb/init-functions

case "${1}" in
   start)
      log_info_msg "Retrying failed uevents, if any..."

      rundir=/run/udev
      # From Debian: "copy the rules generated before / was mounted
      # read-write":

      for file in ${rundir}/tmp-rules--*; do
         dest=${file##*tmp-rules--}
         [ "$dest" = '*' ] && break
         cat $file >> /etc/udev/rules.d/$dest
         rm -f $file
      done

      # Re-trigger the uevents that may have failed,
      # in hope they will succeed now
      /bin/sed -e 's/#.*$//' /etc/sysconfig/udev_retry | /bin/grep -v '^$' | \
      while read line ; do
         for subsystem in $line ; do
            /bin/udevadm trigger --subsystem-match=$subsystem --action=add
         done
      done

      # Now wait for udevd to process the uevents we triggered
```

```
      if ! is_true "$OMIT_UDEV_RETRY_SETTLE"; then
         /bin/udevadm settle
      fi

      evaluate_retval
      ;;

   *)
      echo "Usage ${0} {start}"
      exit 1
      ;;
esac

exit 0

# End udev_retry
```

D.11. /etc/rc.d/init.d/cleanfs

```
#!/bin/sh
########################################################################
# Begin cleanfs
#
# Description : Clean file system
#
# Authors     : Gerard Beekmans - gerard@linuxfromscratch.org
#               DJ Lucas - dj@linuxfromscratch.org
# Update      : Bruce Dubbs - bdubbs@linuxfromscratch.org
#
# Version     : LFS 7.0
#
########################################################################

### BEGIN INIT INFO
# Provides:            cleanfs
# Required-Start:      $local_fs
# Should-Start:
# Required-Stop:
# Should-Stop:
# Default-Start:       S
# Default-Stop:
# Short-Description:   Cleans temporary directories early in the boot process.
# Description:         Cleans temporary directories /run, /var/lock, and
#                      optionally, /tmp.  cleanfs also creates /run/utmp
#                      and any files defined in /etc/sysconfig/createfiles.
# X-LFS-Provided-By:   LFS
### END INIT INFO

. /lib/lsb/init-functions

# Function to create files/directory on boot.
create_files()
{
   # Input to file descriptor 9 and output to stdin (redirection)
   exec 9>&0 < /etc/sysconfig/createfiles

   while read name type perm usr grp dtype maj min junk
   do
      # Ignore comments and blank lines.
      case "${name}" in
         ""|\#*) continue ;;
      esac

      # Ignore existing files.
      if [ ! -e "${name}" ]; then
         # Create stuff based on its type.
         case "${type}" in
```

```
            dir)
               mkdir "${name}"
               ;;
            file)
               :> "${name}"
               ;;
            dev)
               case "${dtype}" in
                  char)
                     mknod "${name}" c ${maj} ${min}
                     ;;
                  block)
                     mknod "${name}" b ${maj} ${min}
                     ;;
                  pipe)
                     mknod "${name}" p
                     ;;
                  *)
                     log_warning_msg "\nUnknown device type: ${dtype}"
                     ;;
               esac
               ;;
            *)
               log_warning_msg "\nUnknown type: ${type}"
               continue
               ;;
         esac

         # Set up the permissions, too.
         chown ${usr}:${grp} "${name}"
         chmod ${perm} "${name}"
      fi
   done

   # Close file descriptor 9 (end redirection)
   exec 0>&9 9>&-
   return 0
}

case "${1}" in
   start)
      log_info_msg "Cleaning file systems:"

      if [ "${SKIPTMPCLEAN}" = "" ]; then
         log_info_msg2 " /tmp"
         cd /tmp &&
         find . -xdev -mindepth 1 ! -name lost+found -delete || failed=1
      fi

      > /run/utmp

      if grep -q '^utmp:' /etc/group ; then
         chmod 664 /run/utmp
         chgrp utmp /run/utmp
      fi

      (exit ${failed})
      evaluate_retval

      if grep -E -qv '^(#|$)' /etc/sysconfig/createfiles 2>/dev/null; then
         log_info_msg "Creating files and directories... "
         create_files      # Always returns 0
         evaluate_retval
      fi

      exit $failed
      ;;
```

```
   *)
      echo "Usage: ${0} {start}"
      exit 1
      ;;
esac

# End cleanfs
```

D.12. /etc/rc.d/init.d/console

```
#!/bin/sh
########################################################################
# Begin console
#
# Description : Sets keymap and screen font
#
# Authors     : Gerard Beekmans - gerard@linuxfromscratch.org
#               Alexander E. Patrakov
#               DJ Lucas - dj@linuxfromscratch.org
# Update      : Bruce Dubbs - bdubbs@linuxfromscratch.org
#
# Version     : LFS 7.0
#
########################################################################

### BEGIN INIT INFO
# Provides:            console
# Required-Start:      $local_fs
# Should-Start:        udev_retry
# Required-Stop:
# Should-Stop:
# Default-Start:       S
# Default-Stop:
# Short-Description:   Sets up a localised console.
# Description:         Sets up fonts and language settings for the user's
#                      local as defined by /etc/sysconfig/console.
# X-LFS-Provided-By:   LFS
### END INIT INFO

. /lib/lsb/init-functions

# Native English speakers probably don't have /etc/sysconfig/console at all
[ -r /etc/sysconfig/console ] && . /etc/sysconfig/console

failed=0

case "${1}" in
   start)
      # See if we need to do anything
      if [ -z "${KEYMAP}"         ] && [ -z "${KEYMAP_CORRECTIONS}" ] &&
         [ -z "${FONT}"           ] && [ -z "${LEGACY_CHARSET}"     ] &&
         ! is_true "${UNICODE}"; then
         exit 0
      fi

      # There should be no bogus failures below this line!
      log_info_msg "Setting up Linux console..."

      # Figure out if a framebuffer console is used
      [ -d /sys/class/graphics/fbcon ] && use_fb=1 || use_fb=0

      # Figure out the command to set the console into the
      # desired mode
      is_true "${UNICODE}" &&
         MODE_COMMAND="echo -en '\033%G' && kbd_mode -u" ||
         MODE_COMMAND="echo -en '\033%@\033(K' && kbd_mode -a"
```

```
      # On framebuffer consoles, font has to be set for each vt in
      # UTF-8 mode. This doesn't hurt in non-UTF-8 mode also.

      ! is_true "${use_fb}" || [ -z "${FONT}" ] ||
         MODE_COMMAND="${MODE_COMMAND} && setfont ${FONT}"

      # Apply that command to all consoles mentioned in
      # /etc/inittab. Important: in the UTF-8 mode this should
      # happen before setfont, otherwise a kernel bug will
      # show up and the unicode map of the font will not be
      # used.

      for TTY in `grep '^[^#].*respawn:/sbin/agetty' /etc/inittab |
         grep -o '\btty[[:digit:]]*\b'`
      do
         openvt -f -w -c ${TTY#tty} -- \
            /bin/sh -c "${MODE_COMMAND}" || failed=1
      done

      # Set the font (if not already set above) and the keymap
      [ "${use_fb}" == "1" ] || [ -z "${FONT}" ] || setfont $FONT || failed=1

      [ -z "${KEYMAP}" ] ||
         loadkeys ${KEYMAP} >/dev/null 2>&1 ||
         failed=1

      [ -z "${KEYMAP_CORRECTIONS}" ] ||
         loadkeys ${KEYMAP_CORRECTIONS} >/dev/null 2>&1 ||
         failed=1

      # Convert the keymap from $LEGACY_CHARSET to UTF-8
      [ -z "$LEGACY_CHARSET" ] ||
         dumpkeys -c "$LEGACY_CHARSET" | loadkeys -u >/dev/null 2>&1 ||
         failed=1

      # If any of the commands above failed, the trap at the
      # top would set $failed to 1
      ( exit $failed )
      evaluate_retval

      exit $failed
      ;;

   *)
      echo "Usage:  ${0} {start}"
      exit 1
      ;;
esac

# End console
```

D.13. /etc/rc.d/init.d/localnet

```
#!/bin/sh
########################################################################
# Begin localnet
#
# Description : Loopback device
#
# Authors     : Gerard Beekmans  - gerard@linuxfromscratch.org
#               DJ Lucas - dj@linuxfromscratch.org
# Update      : Bruce Dubbs - bdubbs@linuxfromscratch.org
#
# Version     : LFS 7.0
#
########################################################################
```

```
### BEGIN INIT INFO
# Provides:            localnet
# Required-Start:      mountvirtfs
# Should-Start:        modules
# Required-Stop:
# Should-Stop:
# Default-Start:       S
# Default-Stop:        0 6
# Short-Description:   Starts the local network.
# Description:         Sets the hostname of the machine and starts the
#                      loopback interface.
# X-LFS-Provided-By:   LFS
### END INIT INFO

. /lib/lsb/init-functions
[ -r /etc/sysconfig/network ] && . /etc/sysconfig/network
[ -r /etc/hostname ] && HOSTNAME=`cat /etc/hostname`

case "${1}" in
   start)
      log_info_msg "Bringing up the loopback interface..."
      ip addr add 127.0.0.1/8 label lo dev lo
      ip link set lo up
      evaluate_retval

      log_info_msg "Setting hostname to ${HOSTNAME}..."
      hostname ${HOSTNAME}
      evaluate_retval
      ;;

   stop)
      log_info_msg "Bringing down the loopback interface..."
      ip link set lo down
      evaluate_retval
      ;;

   restart)
      ${0} stop
      sleep 1
      ${0} start
      ;;

   status)
      echo "Hostname is: $(hostname)"
      ip link show lo
      ;;

   *)
      echo "Usage: ${0} {start|stop|restart|status}"
      exit 1
      ;;
esac

exit 0

# End localnet
```

D.14. /etc/rc.d/init.d/sysctl

```
#!/bin/sh
########################################################################
# Begin sysctl
#
# Description : File uses /etc/sysctl.conf to set kernel runtime
#               parameters
#
# Authors     : Nathan Coulson (nathan@linuxfromscratch.org)
```

```
#               Matthew Burgress (matthew@linuxfromscratch.org)
#               DJ Lucas - dj@linuxfromscratch.org
# Update      : Bruce Dubbs - bdubbs@linuxfromscratch.org
#
# Version     : LFS 7.0
#
###############################################################################

### BEGIN INIT INFO
# Provides:            sysctl
# Required-Start:      mountvirtfs
# Should-Start:        console
# Required-Stop:
# Should-Stop:
# Default-Start:       S
# Default-Stop:
# Short-Description:   Makes changes to the proc filesystem
# Description:         Makes changes to the proc filesystem as defined in
#                      /etc/sysctl.conf.  See 'man sysctl(8)'.
# X-LFS-Provided-By:   LFS
### END INIT INFO

. /lib/lsb/init-functions

case "${1}" in
   start)
      if [ -f "/etc/sysctl.conf" ]; then
         log_info_msg "Setting kernel runtime parameters..."
         sysctl -q -p
         evaluate_retval
      fi
      ;;

   status)
      sysctl -a
      ;;

   *)
      echo "Usage: ${0} {start|status}"
      exit 1
      ;;
esac

exit 0

# End sysctl
```

D.15. /etc/rc.d/init.d/sysklogd

```
#!/bin/sh
###############################################################################
# Begin sysklogd
#
# Description : Sysklogd loader
#
# Authors     : Gerard Beekmans - gerard@linuxfromscratch.org
#               DJ Lucas - dj@linuxfromscratch.org
# Update      : Bruce Dubbs - bdubbs@linuxfromscratch.org
# Update      : Bruce Dubbs - bdubbs@linuxfromscratch.org LFS12.1
#               Remove kernel log daemon.  The functionality has been
#               merged with syslogd.
#
# Version     : LFS 7.0
#
###############################################################################

### BEGIN INIT INFO
```

```
# Provides:            $syslog
# Required-Start:      $first localnet
# Should-Start:
# Required-Stop:       $local_fs
# Should-Stop:         sendsignals
# Default-Start:       2 3 4 5
# Default-Stop:        0 1 6
# Short-Description:   Starts system log daemon.
# Description:         Starts system log daemon.
#                      /etc/fstab.
# X-LFS-Provided-By:   LFS
### END INIT INFO

. /lib/lsb/init-functions

case "${1}" in
   start)
      log_info_msg "Starting system log daemon..."
      parms=${SYSKLOGD_PARMS-'-m 0'}
      start_daemon /sbin/syslogd $parms
      evaluate_retval
      ;;

   stop)
      log_info_msg "Stopping system log daemon..."
      killproc /sbin/syslogd
      evaluate_retval
      ;;

   reload)
      log_info_msg "Reloading system log daemon config file..."
      pid=`pidofproc syslogd`
      kill -HUP "${pid}"
      evaluate_retval
      ;;

   restart)
      ${0} stop
      sleep 1
      ${0} start
      ;;

   status)
      statusproc /sbin/syslogd
      ;;

   *)
      echo "Usage: ${0} {start|stop|reload|restart|status}"
      exit 1
      ;;
esac

exit 0

# End sysklogd
```

D.16. /etc/rc.d/init.d/network

```
#!/bin/sh
########################################################################
# Begin network
#
# Description : Network Control Script
#
# Authors     : Gerard Beekmans - gerard@linuxfromscratch.org
#               Nathan Coulson - nathan@linuxfromscratch.org
#               Kevin P. Fleming - kpfleming@linuxfromscratch.org
```

```
#               DJ Lucas - dj@linuxfromscratch.org
# Update      : Bruce Dubbs - bdubbs@linuxfromscratch.org
#
# Version     : LFS 7.0
#
########################################################################

### BEGIN INIT INFO
# Provides:            $network
# Required-Start:      $local_fs localnet swap
# Should-Start:        $syslog firewalld iptables nftables
# Required-Stop:       $local_fs localnet swap
# Should-Stop:         $syslog firewalld iptables nftables
# Default-Start:       2 3 4 5
# Default-Stop:        0 1 6
# Short-Description:   Starts and configures network interfaces.
# Description:         Starts and configures network interfaces.
# X-LFS-Provided-By:   LFS
### END INIT INFO

case "${1}" in
   start)
      # if the default route exists, network is already configured
      if ip route | grep -q "^default"; then exit 0; fi
      # Start all network interfaces
      for file in /etc/sysconfig/ifconfig.*
      do
         interface=${file##*/ifconfig.}

         # Skip if $file is * (because nothing was found)
         if [ "${interface}" = "*" ]; then continue; fi

         /sbin/ifup ${interface}
      done
      ;;

   stop)
      # Unmount any network mounted file systems
       umount --all --force --types nfs,cifs,nfs4

      # Reverse list
      net_files=""
      for file in  /etc/sysconfig/ifconfig.*
      do
         net_files="${file} ${net_files}"
      done

      # Stop all network interfaces
      for file in ${net_files}
      do
         interface=${file##*/ifconfig.}

         # Skip if $file is * (because nothing was found)
         if [ "${interface}" = "*" ]; then continue; fi

         # See if interface exists
         if [ ! -e /sys/class/net/$interface ]; then continue; fi

         # Is interface UP?
         ip link show $interface 2>/dev/null | grep -q "state UP"
         if [ $? -ne 0 ];  then continue; fi

         /sbin/ifdown ${interface}
      done
      ;;

   restart)
```

```
      ${0} stop
      sleep 1
      ${0} start
      ;;

   *)
      echo "Usage: ${0} {start|stop|restart}"
      exit 1
      ;;
esac

exit 0

# End network
```

D.17. /etc/rc.d/init.d/sendsignals

```
#!/bin/sh
########################################################################
# Begin sendsignals
#
# Description : Sendsignals Script
#
# Authors     : Gerard Beekmans - gerard@linuxfromscratch.org
#               DJ Lucas - dj@linuxfromscratch.org
# Update      : Bruce Dubbs - bdubbs@linuxfromscratch.org
#
# Version     : LFS 7.0
#
########################################################################

### BEGIN INIT INFO
# Provides:            sendsignals
# Required-Start:
# Should-Start:
# Required-Stop:       $local_fs swap localnet
# Should-Stop:
# Default-Start:
# Default-Stop:        0 6
# Short-Description:   Attempts to kill remaining processes.
# Description:         Attempts to kill remaining processes.
# X-LFS-Provided-By:   LFS
### END INIT INFO

. /lib/lsb/init-functions

case "${1}" in
   stop)
      omit=$(pidof mdmon)
      [ -n "$omit" ] && omit="-o $omit"

      log_info_msg "Sending all processes the TERM signal..."
      killall5 -15 $omit
      error_value=${?}

      sleep ${KILLDELAY}

      if [ "${error_value}" = 0 -o "${error_value}" = 2 ]; then
         log_success_msg
      else
         log_failure_msg
      fi

      log_info_msg "Sending all processes the KILL signal..."
      killall5 -9 $omit
      error_value=${?}
```

```
      sleep ${KILLDELAY}

      if [ "${error_value}" = 0 -o "${error_value}" = 2 ]; then
         log_success_msg
      else
         log_failure_msg
      fi
      ;;

   *)
      echo "Usage: ${0} {stop}"
      exit 1
      ;;

esac

exit 0

# End sendsignals
```

D.18. /etc/rc.d/init.d/reboot

```
#!/bin/sh
########################################################################
# Begin reboot
#
# Description : Reboot Scripts
#
# Authors     : Gerard Beekmans - gerard@linuxfromscratch.org
#               DJ Lucas - dj@linuxfromscratch.org
# Updates     : Bruce Dubbs - bdubbs@linuxfromscratch.org
#             : Pierre Labastie - pierre@linuxfromscratch.org
#
# Version     : LFS 7.0
#
# Notes       : Update March 24th, 2022: change "stop" to "start".
#               Add the $last facility to Required-start
#
########################################################################

### BEGIN INIT INFO
# Provides:            reboot
# Required-Start:      $last
# Should-Start:
# Required-Stop:
# Should-Stop:
# Default-Start:       6
# Default-Stop:
# Short-Description:   Reboots the system.
# Description:         Reboots the System.
# X-LFS-Provided-By:   LFS
### END INIT INFO

. /lib/lsb/init-functions

case "${1}" in
   start)
      log_info_msg "Restarting system..."
      reboot -d -f -i
      ;;

   *)
      echo "Usage: ${0} {start}"
      exit 1
      ;;

esac
```

```
# End reboot
```

D.19. /etc/rc.d/init.d/halt

```
#!/bin/sh
########################################################################
# Begin halt
#
# Description : Halt Script
#
# Authors     : Gerard Beekmans - gerard@linuxfromscratch.org
#               DJ Lucas - dj@linuxfromscratch.org
# Update      : Bruce Dubbs - bdubbs@linuxfromscratch.org
#             : Pierre Labastie - pierre@linuxfromscratch.org
#
# Version     : LFS 7.0
#
# Notes       : Update March 24th, 2022: change "stop" to "start".
#               Add the $last facility to Required-start
#
########################################################################

### BEGIN INIT INFO
# Provides:            halt
# Required-Start:      $last
# Should-Start:
# Required-Stop:
# Should-Stop:
# Default-Start:       0
# Default-Stop:
# Short-Description:   Halts the system.
# Description:         Halts the System.
# X-LFS-Provided-By:   LFS
### END INIT INFO

case "${1}" in
   start)
      halt -d -f -i -p
      ;;

   *)
      echo "Usage: {start}"
      exit 1
      ;;
esac

# End halt
```

D.20. /etc/rc.d/init.d/template

```
#!/bin/sh
########################################################################
# Begin scriptname
#
# Description :
#
# Authors     :
#
# Version     : LFS x.x
#
# Notes       :
#
########################################################################
```

```
### BEGIN INIT INFO
# Provides:            template
# Required-Start:
# Should-Start:
# Required-Stop:
# Should-Stop:
# Default-Start:
# Default-Stop:
# Short-Description:
# Description:
# X-LFS-Provided-By:
### END INIT INFO

. /lib/lsb/init-functions

case "${1}" in
   start)
      log_info_msg "Starting..."
    # if it is possible to use start_daemon
      start_daemon fully_qualified_path
    # if it is not possible to use start_daemon
    # (command to start the daemon is not simple enough)
      if ! pidofproc daemon_name_as_reported_by_ps >/dev/null; then
         command_to_start_the_service
      fi
      evaluate_retval
      ;;

   stop)
      log_info_msg "Stopping..."
    # if it is possible to use killproc
      killproc fully_qualified_path
    # if it is not possible to use killproc
    # (the daemon shouldn't be stopped by killing it)
      if pidofproc daemon_name_as_reported_by_ps >/dev/null; then
         command_to_stop_the_service
      fi
      evaluate_retval
      ;;

   restart)
      ${0} stop
      sleep 1
      ${0} start
      ;;

   *)
      echo "Usage: ${0} {start|stop|restart}"
      exit 1
      ;;
esac

exit 0

# End scriptname
```

D.21. /etc/sysconfig/modules

```
#############################################################################
# Begin /etc/sysconfig/modules
#
# Description : Module auto-loading configuration
#
# Authors     :
#
# Version     : 00.00
#
```

```
# Notes       : The syntax of this file is as follows:
#               <module> [<arg1> <arg2> ...]
#
# Each module should be on its own line, and any options that you want
# passed to the module should follow it.  The line deliminator is either
# a space or a tab.
#########################################################################

# End /etc/sysconfig/modules
```

D.22. /etc/sysconfig/createfiles

```
#########################################################################
# Begin /etc/sysconfig/createfiles
#
# Description : Createfiles script config file
#
# Authors     :
#
# Version     : 00.00
#
# Notes       : The syntax of this file is as follows:
#               if type is equal to "file" or "dir"
#                <filename> <type> <permissions> <user> <group>
#               if type is equal to "dev"
#                <filename> <type> <permissions> <user> <group> <devtype>
#             <major> <minor>
#
#               <filename> is the name of the file which is to be created
#               <type> is either file, dir, or dev.
#                       file creates a new file
#                       dir creates a new directory
#                       dev creates a new device
#               <devtype> is either block, char or pipe
#                       block creates a block device
#                       char creates a character device
#                       pipe creates a pipe, this will ignore the <major> and
#           <minor> fields
#               <major> and <minor> are the major and minor numbers used for
#     the device.
#########################################################################

# End /etc/sysconfig/createfiles
```

D.23. /etc/sysconfig/udev-retry

```
#########################################################################
# Begin /etc/sysconfig/udev_retry
#
# Description : udev_retry script configuration
#
# Authors     :
#
# Version     : 00.00
#
# Notes       : Each subsystem that may need to be re-triggered after mountfs
#               runs should be listed in this file.  Probable subsystems to be
#               listed here are rtc (due to /var/lib/hwclock/adjtime) and sound
#               (due to both /var/lib/alsa/asound.state and /usr/sbin/alsactl).
#               Entries are whitespace-separated.
#########################################################################

rtc

# End /etc/sysconfig/udev_retry
```

D.24. /sbin/ifup

```
#!/bin/sh
########################################################################
# Begin /sbin/ifup
#
# Description : Interface Up
#
# Authors     : Nathan Coulson - nathan@linuxfromscratch.org
#               Kevin P. Fleming - kpfleming@linuxfromscratch.org
# Update      : Bruce Dubbs - bdubbs@linuxfromscratch.org
#               DJ Lucas - dj@linuxfromscratch.org
#
# Version     : LFS 7.7
#
# Notes       : The IFCONFIG variable is passed to the SERVICE script
#               in the /lib/services directory, to indicate what file the
#               service should source to get interface specifications.
#
########################################################################

up()
{
  log_info_msg "Bringing up the ${1} interface..."

  if ip link show $1 > /dev/null 2>&1; then
     link_status=`ip link show $1`

     if [ -n "${link_status}" ]; then
        if ! echo "${link_status}" | grep -q UP; then
           ip link set $1 up
        fi
     fi

  else
     log_failure_msg "Interface ${IFACE} doesn't exist."
     exit 1
  fi

  evaluate_retval
}

RELEASE="7.7"

USAGE="Usage: $0 [ -hV ] [--help] [--version] interface"
VERSTR="LFS ifup, version ${RELEASE}"

while [ $# -gt 0 ]; do
   case "$1" in
      --help | -h)     help="y"; break ;;

      --version | -V)  echo "${VERSTR}"; exit 0 ;;

      -*)              echo "ifup: ${1}: invalid option" >&2
                       echo "${USAGE}" >& 2
                       exit 2 ;;

      *)               break ;;
   esac
done

if [ -n "$help" ]; then
   echo "${VERSTR}"
   echo "${USAGE}"
   echo
   cat << HERE_EOF
ifup is used to bring up a network interface.  The interface
```

```
parameter, e.g. eth0 or eth0:2, must match the trailing part of the
interface specifications file, e.g. /etc/sysconfig/ifconfig.eth0:2.

HERE_EOF
   exit 0
fi

file=/etc/sysconfig/ifconfig.${1}

# Skip backup files
[ "${file}" = "${file%""~""}" ] || exit 0

. /lib/lsb/init-functions

if [ ! -r "${file}" ]; then
   log_failure_msg "Unable to bring up ${1} interface! ${file} is missing or cannot be accessed."
   exit 1
fi

.  $file

if [ "$IFACE" = "" ]; then
   log_failure_msg "Unable to bring up ${1} interface! ${file} does not define an interface [IFACE]."
   exit 1
fi

# Do not process this service if started by boot, and ONBOOT
# is not set to yes
if [ "${IN_BOOT}" = "1" -a "${ONBOOT}" != "yes" ]; then
   exit 0
fi

# Bring up the interface
if [ "$VIRTINT" != "yes" ]; then
   up ${IFACE}
fi

for S in ${SERVICE}; do
  if [ ! -x "/lib/services/${S}" ]; then
    MSG="\nUnable to process ${file}.  Either "
    MSG="${MSG}the SERVICE '${S} was not present "
    MSG="${MSG}or cannot be executed."
    log_failure_msg "$MSG"
    exit 1
  fi
done

#if [ "${SERVICE}" = "wpa" ]; then log_success_msg; fi

# Create/configure the interface
for S in ${SERVICE}; do
  IFCONFIG=${file} /lib/services/${S} ${IFACE} up
done

# Set link up virtual interfaces
if [ "${VIRTINT}" == "yes" ]; then
   up ${IFACE}
fi

# Bring up any additional interface components
for I in $INTERFACE_COMPONENTS; do up $I; done

# Set MTU if requested. Check if MTU has a "good" value.
if test -n "${MTU}"; then
   if [[ ${MTU} =~ ^[0-9]+$ ]] && [[ $MTU -ge 68 ]] ; then
      for I in $IFACE $INTERFACE_COMPONENTS; do
         ip link set dev $I mtu $MTU;
```

```
      done
   else
      log_info_msg2 "Invalid MTU $MTU"
   fi
fi

# Set the route default gateway if requested
if [ -n "${GATEWAY}" ]; then
   if ip route | grep -q default; then
      log_warning_msg "Gateway already setup; skipping."
   else
      log_info_msg "Adding default gateway ${GATEWAY} to the ${IFACE} interface..."
      ip route add default via ${GATEWAY} dev ${IFACE}
      evaluate_retval
   fi
fi

# End /sbin/ifup
```

D.25. /sbin/ifdown

```
#!/bin/bash
########################################################################
# Begin /sbin/ifdown
#
# Description : Interface Down
#
# Authors     : Nathan Coulson - nathan@linuxfromscratch.org
#               Kevin P. Fleming - kpfleming@linuxfromscratch.org
# Update      : Bruce Dubbs - bdubbs@linuxfromscratch.org
#
# Version     : LFS 7.0
#
# Notes       : the IFCONFIG variable is passed to the scripts found
#               in the /lib/services directory, to indicate what file the
#               service should source to get interface specifications.
#
########################################################################

RELEASE="7.0"

USAGE="Usage: $0 [ -hV ] [--help] [--version] interface"
VERSTR="LFS ifdown, version ${RELEASE}"

while [ $# -gt 0 ]; do
   case "$1" in
      --help | -h)     help="y"; break ;;

      --version | -V)  echo "${VERSTR}"; exit 0 ;;

      -*)              echo "ifup: ${1}: invalid option" >&2
                       echo "${USAGE}" >& 2
                       exit 2 ;;

      *)               break ;;
   esac
done

if [ -n "$help" ]; then
   echo "${VERSTR}"
   echo "${USAGE}"
   echo
   cat << HERE_EOF
ifdown is used to bring down a network interface.  The interface
parameter, e.g. eth0 or eth0:2, must match the trailing part of the
interface specifications file, e.g. /etc/sysconfig/ifconfig.eth0:2.
```

```
HERE_EOF
   exit 0
fi

file=/etc/sysconfig/ifconfig.${1}

# Skip backup files
[ "${file}" = "${file%""~""}" ] || exit 0

. /lib/lsb/init-functions

if [ ! -r "${file}" ]; then
   log_warning_msg "${file} is missing or cannot be accessed."
   exit 1
fi

. ${file}

if [ "$IFACE" = "" ]; then
   log_failure_msg "${file} does not define an interface [IFACE]."
   exit 1
fi

# We only need to first service to bring down the interface
S=`echo ${SERVICE} | cut -f1 -d" "`

if ip link show ${IFACE} > /dev/null 2>&1; then
   if [ -n "${S}" -a -x "/lib/services/${S}" ]; then
     IFCONFIG=${file} /lib/services/${S} ${IFACE} down
   else
     MSG="Unable to process ${file}.  Either "
     MSG="${MSG}the SERVICE variable was not set "
     MSG="${MSG}or the specified service cannot be executed."
     log_failure_msg "$MSG"
     exit 1
  fi
else
   log_warning_msg "Interface ${1} doesn't exist."
fi

# Leave the interface up if there are additional interfaces in the device
link_status=`ip link show ${IFACE} 2>/dev/null`

if [ -n "${link_status}" ]; then
   if [ "$(echo "${link_status}" | grep UP)" != "" ]; then
      if [ "$(ip addr show ${IFACE} | grep 'inet ')" == ""  ]; then
         log_info_msg "Bringing down the ${IFACE} interface..."
         ip link set ${IFACE} down
         evaluate_retval
      fi
   fi
fi

# End /sbin/ifdown
```

D.26. /lib/services/ipv4-static

```
#!/bin/sh
########################################################################
# Begin /lib/services/ipv4-static
#
# Description : IPV4 Static Boot Script
#
# Authors     : Nathan Coulson - nathan@linuxfromscratch.org
#               Kevin P. Fleming - kpfleming@linuxfromscratch.org
# Update      : Bruce Dubbs - bdubbs@linuxfromscratch.org
#
```

```
# Version     : LFS 7.0
#
########################################################################

. /lib/lsb/init-functions
. ${IFCONFIG}

if [ -z "${IP}" ]; then
   log_failure_msg "\nIP variable missing from ${IFCONFIG}, cannot continue."
   exit 1
fi

if [ -z "${PREFIX}" -a -z "${PEER}" ]; then
   log_warning_msg "\nPREFIX variable missing from ${IFCONFIG}, assuming 24."
   PREFIX=24
   args="${args} ${IP}/${PREFIX}"

elif [ -n "${PREFIX}" -a -n "${PEER}" ]; then
   log_failure_msg "\nPREFIX and PEER both specified in ${IFCONFIG}, cannot continue."
   exit 1

elif [ -n "${PREFIX}" ]; then
   args="${args} ${IP}/${PREFIX}"

elif [ -n "${PEER}" ]; then
   args="${args} ${IP} peer ${PEER}"
fi

if [ -n "${LABEL}" ]; then
   args="${args} label ${LABEL}"
fi

if [ -n "${BROADCAST}" ]; then
   args="${args} broadcast ${BROADCAST}"
fi

case "${2}" in
   up)
      if [ "$(ip addr show ${1} 2>/dev/null | grep ${IP}/)" = "" ]; then
         log_info_msg "Adding IPv4 address ${IP} to the ${1} interface..."
         ip addr add ${args} dev ${1}
         evaluate_retval
      else
         log_warning_msg "Cannot add IPv4 address ${IP} to ${1}.  Already present."
      fi
   ;;

   down)
      if [ "$(ip addr show ${1} 2>/dev/null | grep ${IP}/)" != "" ]; then
         log_info_msg "Removing IPv4 address ${IP} from the ${1} interface..."
         ip addr del ${args} dev ${1}
         evaluate_retval
      fi

      if [ -n "${GATEWAY}" ]; then
         # Only remove the gateway if there are no remaining ipv4 addresses
         if [ "$(ip addr show ${1} 2>/dev/null | grep 'inet ')" != "" ]; then
            log_info_msg "Removing default gateway..."
            ip route del default
            evaluate_retval
         fi
      fi
   ;;

   *)
      echo "Usage: ${0} [interface] {up|down}"
      exit 1
```

```
   ;;
esac

# End /lib/services/ipv4-static
```

D.27. /lib/services/ipv4-static-route

```
#!/bin/sh
########################################################################
# Begin /lib/services/ipv4-static-route
#
# Description : IPV4 Static Route Script
#
# Authors     : Kevin P. Fleming - kpfleming@linuxfromscratch.org
#               DJ Lucas - dj@linuxfromscratch.org
# Update      : Bruce Dubbs - bdubbs@linuxfromscratch.org
#
# Version     : LFS 7.0
#
########################################################################

. /lib/lsb/init-functions
. ${IFCONFIG}

case "${TYPE}" in
   ("" | "network")
      need_ip=1
      need_gateway=1
   ;;

   ("default")
      need_gateway=1
      args="${args} default"
      desc="default"
   ;;

   ("host")
      need_ip=1
   ;;

   ("unreachable")
      need_ip=1
      args="${args} unreachable"
      desc="unreachable "
   ;;

   (*)
      log_failure_msg "Unknown route type (${TYPE}) in ${IFCONFIG}, cannot continue."
      exit 1
   ;;
esac

if [ -n "${GATEWAY}" ]; then
   MSG="The GATEWAY variable cannot be set in ${IFCONFIG} for static routes.\n"
   log_failure_msg "$MSG Use STATIC_GATEWAY only, cannot continue"
   exit 1
fi

if [ -n "${need_ip}" ]; then
   if [ -z "${IP}" ]; then
      log_failure_msg "IP variable missing from ${IFCONFIG}, cannot continue."
      exit 1
   fi

   if [ -z "${PREFIX}" ]; then
      log_failure_msg "PREFIX variable missing from ${IFCONFIG}, cannot continue."
      exit 1
```

```
   fi

   args="${args} ${IP}/${PREFIX}"
   desc="${desc}${IP}/${PREFIX}"
fi

if [ -n "${need_gateway}" ]; then
   if [ -z "${STATIC_GATEWAY}" ]; then
      log_failure_msg "STATIC_GATEWAY variable missing from ${IFCONFIG}, cannot continue."
      exit 1
   fi
   args="${args} via ${STATIC_GATEWAY}"
fi

if [ -n "${SOURCE}" ]; then
        args="${args} src ${SOURCE}"
fi

case "${2}" in
   up)
      log_info_msg "Adding '${desc}' route to the ${1} interface..."
      ip route add ${args} dev ${1}
      evaluate_retval
   ;;

   down)
      log_info_msg "Removing '${desc}' route from the ${1} interface..."
      ip route del ${args} dev ${1}
      evaluate_retval
   ;;

   *)
      echo "Usage: ${0} [interface] {up|down}"
      exit 1
   ;;
esac

# End /lib/services/ipv4-static-route
```

Annexe E. Règles de configuration d'Udev

Les règles de cette annexe sont listées pour votre commodité. Leur installation se fait en principe via les instructions du Section 8.76, « Udev de Systemd-257.8. »

E.1. 55-lfs.rules

```
# /etc/udev/rules.d/55-lfs.rules: Rule definitions for LFS.

# Core kernel devices

# This causes the system clock to be set as soon as /dev/rtc becomes available.
SUBSYSTEM=="rtc", ACTION=="add", MODE="0644", RUN+="/etc/rc.d/init.d/setclock start"
KERNEL=="rtc", ACTION=="add", MODE="0644", RUN+="/etc/rc.d/init.d/setclock start"
```

Annexe F. Licences LFS

Ce livre est couvert par la licence Creative Commons Attribution-NonCommercial-ShareAlike 2.0.

Les instructions destinées à l'ordinateur peuvent être extraites selon les termes de la licence MIT.

F.1. Creative Commons License

Creative Commons Legal Code

Attribution-NonCommercial-ShareAlike 2.0

Important

CREATIVE COMMONS CORPORATION IS NOT A LAW FIRM AND DOES NOT PROVIDE LEGAL SERVICES. DISTRIBUTION OF THIS LICENSE DOES NOT CREATE AN ATTORNEY-CLIENT RELATIONSHIP. CREATIVE COMMONS PROVIDES THIS INFORMATION ON AN "AS-IS" BASIS. CREATIVE COMMONS MAKES NO WARRANTIES REGARDING THE INFORMATION PROVIDED, AND DISCLAIMS LIABILITY FOR DAMAGES RESULTING FROM ITS USE.

License

THE WORK (AS DEFINED BELOW) IS PROVIDED UNDER THE TERMS OF THIS CREATIVE COMMONS PUBLIC LICENSE ("CCPL" OR "LICENSE"). THE WORK IS PROTECTED BY COPYRIGHT AND/OR OTHER APPLICABLE LAW. ANY USE OF THE WORK OTHER THAN AS AUTHORIZED UNDER THIS LICENSE OR COPYRIGHT LAW IS PROHIBITED.

BY EXERCISING ANY RIGHTS TO THE WORK PROVIDED HERE, YOU ACCEPT AND AGREE TO BE BOUND BY THE TERMS OF THIS LICENSE. THE LICENSOR GRANTS YOU THE RIGHTS CONTAINED HERE IN CONSIDERATION OF YOUR ACCEPTANCE OF SUCH TERMS AND CONDITIONS.

1. Definitions

   a. "Collective Work" means a work, such as a periodical issue, anthology or encyclopedia, in which the Work in its entirety in unmodified form, along with a number of other contributions, constituting separate and independent works in themselves, are assembled into a collective whole. A work that constitutes a Collective Work will not be considered a Derivative Work (as defined below) for the purposes of this License.

   b. "Derivative Work" means a work based upon the Work or upon the Work and other pre-existing works, such as a translation, musical arrangement, dramatization, fictionalization, motion picture version, sound recording, art reproduction, abridgment, condensation, or any other form in which the Work may be recast, transformed, or adapted, except that a work that constitutes a Collective Work will not be considered a Derivative Work for the purpose of this License. For the avoidance of doubt, where the Work is a musical composition or sound recording, the synchronization of the Work in timed-relation with a moving image ("synching") will be considered a Derivative Work for the purpose of this License.

   c. "Licensor" means the individual or entity that offers the Work under the terms of this License.

   d. "Original Author" means the individual or entity who created the Work.

   e. "Work" means the copyrightable work of authorship offered under the terms of this License.

   f. "You" means an individual or entity exercising rights under this License who has not previously violated the terms of this License with respect to the Work, or who has received express permission from the Licensor to exercise rights under this License despite a previous violation.

g. "License Elements" means the following high-level license attributes as selected by Licensor and indicated in the title of this License: Attribution, Noncommercial, ShareAlike.

2. Fair Use Rights. Nothing in this license is intended to reduce, limit, or restrict any rights arising from fair use, first sale or other limitations on the exclusive rights of the copyright owner under copyright law or other applicable laws.

3. License Grant. Subject to the terms and conditions of this License, Licensor hereby grants You a worldwide, royalty-free, non-exclusive, perpetual (for the duration of the applicable copyright) license to exercise the rights in the Work as stated below:

   a. to reproduce the Work, to incorporate the Work into one or more Collective Works, and to reproduce the Work as incorporated in the Collective Works;

   b. to create and reproduce Derivative Works;

   c. to distribute copies or phonorecords of, display publicly, perform publicly, and perform publicly by means of a digital audio transmission the Work including as incorporated in Collective Works;

   d. to distribute copies or phonorecords of, display publicly, perform publicly, and perform publicly by means of a digital audio transmission Derivative Works;

   The above rights may be exercised in all media and formats whether now known or hereafter devised. The above rights include the right to make such modifications as are technically necessary to exercise the rights in other media and formats. All rights not expressly granted by Licensor are hereby reserved, including but not limited to the rights set forth in Sections 4(e) and 4(f).

4. Restrictions.The license granted in Section 3 above is expressly made subject to and limited by the following restrictions:

   a. You may distribute, publicly display, publicly perform, or publicly digitally perform the Work only under the terms of this License, and You must include a copy of, or the Uniform Resource Identifier for, this License with every copy or phonorecord of the Work You distribute, publicly display, publicly perform, or publicly digitally perform. You may not offer or impose any terms on the Work that alter or restrict the terms of this License or the recipients' exercise of the rights granted hereunder. You may not sublicense the Work. You must keep intact all notices that refer to this License and to the disclaimer of warranties. You may not distribute, publicly display, publicly perform, or publicly digitally perform the Work with any technological measures that control access or use of the Work in a manner inconsistent with the terms of this License Agreement. The above applies to the Work as incorporated in a Collective Work, but this does not require the Collective Work apart from the Work itself to be made subject to the terms of this License. If You create a Collective Work, upon notice from any Licensor You must, to the extent practicable, remove from the Collective Work any reference to such Licensor or the Original Author, as requested. If You create a Derivative Work, upon notice from any Licensor You must, to the extent practicable, remove from the Derivative Work any reference to such Licensor or the Original Author, as requested.

   b. You may distribute, publicly display, publicly perform, or publicly digitally perform a Derivative Work only under the terms of this License, a later version of this License with the same License Elements as this License, or a Creative Commons iCommons license that contains the same License Elements as this License (e.g. Attribution-NonCommercial-ShareAlike 2.0 Japan). You must include a copy of, or the Uniform Resource Identifier for, this License or other license specified in the previous sentence with every copy or phonorecord of each Derivative Work You distribute, publicly display, publicly perform, or publicly digitally perform. You may not offer or impose any terms on the Derivative Works that alter or restrict the terms of this License or the recipients' exercise of the rights granted hereunder, and You must keep intact all notices that refer to this License and to the disclaimer of warranties. You may not distribute, publicly display, publicly perform, or publicly digitally perform the Derivative Work with any technological measures that control access or use

of the Work in a manner inconsistent with the terms of this License Agreement. The above applies to the Derivative Work as incorporated in a Collective Work, but this does not require the Collective Work apart from the Derivative Work itself to be made subject to the terms of this License.

c. You may not exercise any of the rights granted to You in Section 3 above in any manner that is primarily intended for or directed toward commercial advantage or private monetary compensation. The exchange of the Work for other copyrighted works by means of digital file-sharing or otherwise shall not be considered to be intended for or directed toward commercial advantage or private monetary compensation, provided there is no payment of any monetary compensation in connection with the exchange of copyrighted works.

d. If you distribute, publicly display, publicly perform, or publicly digitally perform the Work or any Derivative Works or Collective Works, You must keep intact all copyright notices for the Work and give the Original Author credit reasonable to the medium or means You are utilizing by conveying the name (or pseudonym if applicable) of the Original Author if supplied; the title of the Work if supplied; to the extent reasonably practicable, the Uniform Resource Identifier, if any, that Licensor specifies to be associated with the Work, unless such URI does not refer to the copyright notice or licensing information for the Work; and in the case of a Derivative Work, a credit identifying the use of the Work in the Derivative Work (e.g., "French translation of the Work by Original Author," or "Screenplay based on original Work by Original Author"). Such credit may be implemented in any reasonable manner; provided, however, that in the case of a Derivative Work or Collective Work, at a minimum such credit will appear where any other comparable authorship credit appears and in a manner at least as prominent as such other comparable authorship credit.

e. For the avoidance of doubt, where the Work is a musical composition:

   i. Performance Royalties Under Blanket Licenses. Licensor reserves the exclusive right to collect, whether individually or via a performance rights society (e.g. ASCAP, BMI, SESAC), royalties for the public performance or public digital performance (e.g. webcast) of the Work if that performance is primarily intended for or directed toward commercial advantage or private monetary compensation.

   ii. Mechanical Rights and Statutory Royalties. Licensor reserves the exclusive right to collect, whether individually or via a music rights agency or designated agent (e.g. Harry Fox Agency), royalties for any phonorecord You create from the Work ("cover version") and distribute, subject to the compulsory license created by 17 USC Section 115 of the US Copyright Act (or the equivalent in other jurisdictions), if Your distribution of such cover version is primarily intended for or directed toward commercial advantage or private monetary compensation. 6. Webcasting Rights and Statutory Royalties. For the avoidance of doubt, where the Work is a sound recording, Licensor reserves the exclusive right to collect, whether individually or via a performance-rights society (e.g. SoundExchange), royalties for the public digital performance (e.g. webcast) of the Work, subject to the compulsory license created by 17 USC Section 114 of the US Copyright Act (or the equivalent in other jurisdictions), if Your public digital performance is primarily intended for or directed toward commercial advantage or private monetary compensation.

f. Webcasting Rights and Statutory Royalties. For the avoidance of doubt, where the Work is a sound recording, Licensor reserves the exclusive right to collect, whether individually or via a performance-rights society (e.g. SoundExchange), royalties for the public digital performance (e.g. webcast) of the Work, subject to the compulsory license created by 17 USC Section 114 of the US Copyright Act (or the equivalent in other jurisdictions), if Your public digital performance is primarily intended for or directed toward commercial advantage or private monetary compensation.

5. Representations, Warranties and Disclaimer

UNLESS OTHERWISE MUTUALLY AGREED TO BY THE PARTIES IN WRITING, LICENSOR OFFERS THE WORK AS-IS AND MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND CONCERNING THE WORK, EXPRESS, IMPLIED, STATUTORY OR OTHERWISE, INCLUDING, WITHOUT LIMITATION, WARRANTIES OF TITLE, MERCHANTIBILITY, FITNESS FOR A PARTICULAR PURPOSE, NONINFRINGEMENT, OR THE ABSENCE OF LATENT OR

OTHER DEFECTS, ACCURACY, OR THE PRESENCE OF ABSENCE OF ERRORS, WHETHER OR NOT DISCOVERABLE. SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OF IMPLIED WARRANTIES, SO SUCH EXCLUSION MAY NOT APPLY TO YOU.

6. Limitation on Liability. EXCEPT TO THE EXTENT REQUIRED BY APPLICABLE LAW, IN NO EVENT WILL LICENSOR BE LIABLE TO YOU ON ANY LEGAL THEORY FOR ANY SPECIAL, INCIDENTAL, CONSEQUENTIAL, PUNITIVE OR EXEMPLARY DAMAGES ARISING OUT OF THIS LICENSE OR THE USE OF THE WORK, EVEN IF LICENSOR HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

7. Termination

   a. This License and the rights granted hereunder will terminate automatically upon any breach by You of the terms of this License. Individuals or entities who have received Derivative Works or Collective Works from You under this License, however, will not have their licenses terminated provided such individuals or entities remain in full compliance with those licenses. Sections 1, 2, 5, 6, 7, and 8 will survive any termination of this License.

   b. Subject to the above terms and conditions, the license granted here is perpetual (for the duration of the applicable copyright in the Work). Notwithstanding the above, Licensor reserves the right to release the Work under different license terms or to stop distributing the Work at any time; provided, however that any such election will not serve to withdraw this License (or any other license that has been, or is required to be, granted under the terms of this License), and this License will continue in full force and effect unless terminated as stated above.

8. Miscellaneous

   a. Each time You distribute or publicly digitally perform the Work or a Collective Work, the Licensor offers to the recipient a license to the Work on the same terms and conditions as the license granted to You under this License.

   b. Each time You distribute or publicly digitally perform a Derivative Work, Licensor offers to the recipient a license to the original Work on the same terms and conditions as the license granted to You under this License.

   c. If any provision of this License is invalid or unenforceable under applicable law, it shall not affect the validity or enforceability of the remainder of the terms of this License, and without further action by the parties to this agreement, such provision shall be reformed to the minimum extent necessary to make such provision valid and enforceable.

   d. No term or provision of this License shall be deemed waived and no breach consented to unless such waiver or consent shall be in writing and signed by the party to be charged with such waiver or consent.

   e. This License constitutes the entire agreement between the parties with respect to the Work licensed here. There are no understandings, agreements or representations with respect to the Work not specified here. Licensor shall not be bound by any additional provisions that may appear in any communication from You. This License may not be modified without the mutual written agreement of the Licensor and You.

Important

Creative Commons is not a party to this License, and makes no warranty whatsoever in connection with the Work. Creative Commons will not be liable to You or any party on any legal theory for any damages whatsoever, including without limitation any general, special, incidental or consequential damages arising in connection to this license. Notwithstanding the foregoing two (2) sentences, if Creative Commons has expressly identified itself as the Licensor hereunder, it shall have all rights and obligations of Licensor.

Except for the limited purpose of indicating to the public that the Work is licensed under the CCPL, neither party will use the trademark "Creative Commons" or any related trademark or logo of Creative Commons without the prior written consent of Creative Commons. Any permitted use will be in compliance with Creative Commons' then-current trademark usage guidelines, as may be published on its website or otherwise made available upon request from time to time.

Creative Commons may be contacted at *http://creativecommons.org/*.

F.2. The MIT License

Index

Paquets

Programmes

Bibliothèques

Scripts

Autres

/etc/udev: 221, 223
/etc/udev/hwdb.bin: 223
/etc/vimrc: 217
/run/utmp: 83
/usr/include/asm-generic/*.h: 53, 53
/usr/include/asm/*.h: 53, 53
/usr/include/drm/*.h: 53, 53
/usr/include/linux/*.h: 53, 53
/usr/include/misc/*.h: 53, 53
/usr/include/mtd/*.h: 53, 53
/usr/include/rdma/*.h: 53, 53
/usr/include/scsi/*.h: 53, 53
/usr/include/sound/*.h: 53, 53
/usr/include/video/*.h: 53, 53
/usr/include/xen/*.h: 53, 53
/var/log/btmp: 83
/var/log/lastlog: 83
/var/log/wtmp: 83
/etc/shells: 265
man pages: 101, 101