Лінукс з початків
Версія 7.2

Створено Gerard Beekmans
Відредаговано Matthew Burgess і Bruce Dubbs

Лінукс з початків: версія 7.2

Створено Gerard Beekmans і відредаговано Matthew Burgess і Bruce Dubbs

Linux From Scratch

Version 7.2

Created by Gerard Beekmans
Edited by Matthew Burgess and Bruce Dubbs

Linux From Scratch: Version 7.2

by Created by Gerard Beekmans and Edited by Matthew Burgess and Bruce Dubbs

Вміст

Передмова

Вступ

Моя подорож для вивчення і кращого розуміння Linux почалась десятиліттям назад у 1998. Я щойно встановив мій перший Linux дистрибутив і швидко став заінтригований концепцією і філософією, котра стоїть за Linux.

Завжди є багато шляхів для вирішення одної задачі. Це ж можна сказати про Linux дистрибутиви. Велика кількість існувала протягом років. Деякі існують і до сьогодні, деякі перетворились на щось інше, ще інші відійшли у нашу пам'ять. Усі вони робили свою справу по-різному для вирішення задач їхньої цільової аудиторії. Через те , що існує багато шляхів вирішення одної задачі, Я почав розуміти, що не повинен бути обмеженим якою-небудь реалізацією. Перед тим, як відкрити для себе Лінукс, ми просто погоджувалися з проблемами інших операційних систем, так як не мали іншого вибору. Це було тим, чим було, чи любили ви це, чи ні. З Лінуксом, концепція вибору почала втілюватися. Якщо ти чогось не любив, ти був вільним, навіть заохочуваним, щоб це змінити.

Я спробував число дистрибутивів і не зміг зупинитись на якому-небудь. Це були хороші системи. Це перестало більше бути правильним чи неправильним. Це стало бути справою індивідуального смаку. З цими усіма можливостями вибору, стало очевидним, що не буде одної системи, яка б була ідеальною для мене. Отож, Я вирішив створити мою власну Лінукс систему, яка буде відповідати моїм персональним вимогам.

Щоб по-справжньому зробити її моєю власною системою, Я вирішив скомпілювати все з вихідного коду, замість того щоб використовувати попередньо скомпільовані бінарні пакети. Ця “ідеальна” Лінукс система мала мати переваги інших систем без наслідування їх слабких сторін. На початок, ідея була досить складною. Я як і раніше був прихильним до ідеї, що така система все-таки може бути зібрана.

Після вирішення деяких проблем, такі як кругові залежності і помилки компілювання, Я в кінці кінців зібрав свою Лінукс систему. Це була повністю функціональна і ідеальна для користування як і інші Лінукс системи на той час. Але це було моє творіння. Це було велике задоволення зібрати таку систему самостійно. Єдина річ, яка була б кращою — це створити усі частини системи самостійно. Це була наступна хороша ідея.

Так як Я ділився своїми ідеями і досвідом з іншими членами Лінукс-общини, стало зрозуміло що була велика зацікавленість у них. Стало очевидним, що зібрані користувачами Лінукс системи відповідали не тільки специфічним для користувачів вимогам, а й служили як ідеальна можливість для навчальна програмістів і системних адміністраторів для розвитку їхніх (вже набутих) навичок у роботі з Лінуксом. З цього поширеного зацікавлення, народився проект Лінукс З Початків (Linux From Scratch Projcet, LFS project).

Книга Лінукс З Початків є центральним ядром проекту. Вона надає базу і інструкції які необхідні для Вас, щоб створити вашу власну систему. Поки дана книга надає шаблон який стане результатом системи, яка буде правильно працювати, ви є вільні змінювати інструкції відповідно до себе, що є, зокрема, важливою частиною проекту. Ви контролюєте усе; ми тільки подаємо вам допоміжну руку щоб дати вам поштовх у власну подорож.

Я щиро сподіваюся, що ви будете добре проводити час, працюючи над вашо власною Системою З Початків, і отримаєте задоволення від багатьох переваг системи, яка по-справжньому є вашою особистою.

--
Жерар Бікманс (Gerard Beekmans)
gerard@linuxfromscratch.org

Аудиторія

Існує багато причин чому б ви хотіли прочитати цю книгу. Одне з питань, яке виникає у багатьох людей: “чому йти крізь надокучливе ручне збирання Лінукс системи з початку, коли ти можеш просто завантажити і встановити на свій комп'ютер вже готову?”.

Одна важлива причина існування цього проекту в тому, щоб допомогти вам вивчити роботу Лінукс системи з середини. Створення LFS системи допомагає демонструвати, що ж змушує Лінукс працювати, і як усі речі працюють разом і залежать один від одного. Одна з найкращих якостей, що саме цей набутий досвід допоможе вам забезпечити, є можливість налаштовування Лінукс системи, для вирішення ваших унікальних потреб.

Іншою ключовою перевагою LFS є те, що він дозволяє вам мати більший контроль над системою, не покладаючись на якісь інші реалізації Лінукс. Разом з LFS, ви є на місті водія і диктуєте кожен аспект системи.

LFS дозволяє вам створювати дуже компактні Лінукс системи. Коли ви встановлюєте звичайні дистрибутиви, ви часто змушені встановлювати велику кількість програм, які, можливо, ніколи не будуть використовуватись або тяжкі для розуміння. Ці програми марнують ресурси. Ви можете погодитись, що з сьогоднішніми жорсткими дисками і процесорами вони більше не піддаються строгому моніторингу. Але, інколи, ви все-одно є обмежені в розмірах, якщо не в ще яких-небудь ресурсах. Подумайте про завантажувальні диски, USB накопичувачі і вбудовані системи. Це є області де LFS є корисним.

Іншою перевагою персонально створених Лінукс систем є безпека. Компілюючи цілу систему з вихідного коду, ви є уповноважені вести аудит усього і застосовувати усі створені патчі безпеки. Це не є більше необхідним чекати поки хтось інший скомпілює бінарний пакет , який виправить діру в безпеці. Якщо ви не перевірите і не впровадите його самостійно, ви не маєте гарантій, що новий бінарний пакет зібраний належним чином і адекватно поправляє проблему.

Метою LFS є створення повністю функціональну систему. Якщо ви не бажаєте зібрати свою власну систему Лінукс з вихідного коду, ви не знайдете нічого корисного у інформації з цієї книги.

Є ще багато інших хороших причин щоб створити вашу власну систему LFS, які можуть бути записані тут. Але в кінці кінців, освіта є найбільш сильною. Якщо ви будете надалі отримувати досвід у LFS, ви відкриєте ту силу, яку інформація і знання насправді дають.

Цільові архітектури LFS

Головною цільовою архітектурою LFS є 32-бітний процесор Intel (32-bit Intel CPU). Якщо ви ще не збирали LFS систему раніше, ви повинні, ймовірно, почати з цієї архітектури. 32-бітна архітектура є найбільш широко підтримуваною Лінукс системою і є найбільш сумісною з відкритими і платними програмними продуктами.

З іншого боку, відомо, що інструкції у цій книзі працюють, з деякими модифікаціями, на архітектурах PowerPC і 64-бітними AMD/Intel процесорами. Основною умовою щоб збудувати систему, що використовує один з цих процесорів, є існуюча Лінукс система, наприклад, та, яка була встановлена до LFS системи (Ubuntu, Red Hat/Fedora, SuSE або інший дистрибутив) і підтримує ту архітектуру, яку ви маєте. Також зверніть увагу що 32-бітні дистрибутиви можуть бути встановлені як основні системи на 64-бітних AMD/Intel комп'ютерах.

Деякі інші факти про 64-бітні системи повинні бути тут додані. Порівнюючи з 32-бітною архітектурою,

розміри виконуваних файлів є дещо більшим, а виконання є дещо швидшим. Для прикладу, в тестовій компіляції LFS-6.5 на процесорах Core2Duo, була виміряна дана статистика:

Архітектура	Час компілювання	Розмір
32-бітна	198.5 хвилин	648 Мбайт
64-бітна	190.6 хвилин	709 Мбайт

Як ви можете бачити, 64-бітна архітектура є лиш на 4% швидша і на 9% більша ніж 32-бітна. Виграш від переходу до 64-бітної системи є відносно мінімальним. Звичайно, якщо ви маєте більше ніж 4 Гбайти RAM, або хочете маніпулювати даними які перевищують 4 Гбайти, переваги 64-бітної системи є істотний.

За замовчуванням 64-бітна система яка є результатом LFS, вважається “чистою” 64-бітною системою. Це означає, що вона підтримує тільки 64-бітні виконувані файли. Будуючи “багато бібліотечну” систему вимагає компілювання багатьох програм двічі: перший раз для 32-бітної системи і другий — для 64-бітної. Це не підтримується у книзі LFS тому, що це вимагало б додаткових інструкцій, відмінних від необхідних для побудови простої базової системи Лінукс. Ви можете звернутися до проекту Cross Linux From Scratch (CLFS — кросплатформенний Лінукс з початків) для цієї складної теми.

Останній коментар відносно 64-бітної системи. Є програми, які не можуть бути коректно скомпільовані для “чистої” 64-бітної системи або вимагають спеціальних інструкцій для компіляції. Зазвичай, ці програми мають деякі, специфічні для 32-бітної системи, вставки асемблерною мовою, які будуть мати провал у 64-бітних системах. Включаючи драйвери Xorg з проекту Beyond Linux From Scratch (BLFS). Багато з цих проблем можуть бути вирішеними, але можуть вимагати виконання спеціальних процедур або використання патчей.

LFS і стандарти

Структура LFS відповідає Лінукс стандартам найближче, як це можливо. Основними стандартами є:

- *POSIX.1-2008 (Portable Operating System Interface for uniX — Портативний Стандарт Інтерфейсу Операційних Систем для Юнікс)*;
- *Filesystem Hierarchy Standard (FHS — Стандарт Ієрархії Файлової Системи);*
- *Linux Standard Base (LSB) Core Specification 4.0 (Базові Стандарти Лінукс (LSB) Основні Специфікації 4.0).*

Стандарт LSB має п'ять окремих стандартів: Core (Основні), C++, Desktop (Робочий стіл або візуальна оболонка), Runtime Languages (Скриптові мови) і Printing (Друкування). На додачу до основних вимог є також архітектурно залежні вимоги. LFS намагається відповідати архітектурам, які обговорювалися в попередніх розділах.

> Увага
>
> Багато людей не погоджуються з вимогами стандарту LSB. Основна ціль його визначення є забезпечити, що платні програмні продукти зможуть бути встановлені і виконуватися правильно на сумісних системах. Через те, що LFS є основаним на вихідних кодах програм, користувач має повний контроль над тим, який пакет є бажаним і багато хто вибирає не встановлювати деякі пакети які є визначений LSB.

Створена повна система LFS є здатною до проходження LSB сертифікаційних тестів, але не без багатьох додаткових пакетів які є за сферою LFS. Ці додаткові пакети мають інструкції для встановлення у BLFS проекті.

Пакети які поставляються LFS і є необхідними для задоволення вимог LSB

LSB Core:	Bash, Binutils, Coreutils, Diffutils, File, Findutils, Gawk, Grep, Gzip, M4, Man-DB, Ncurses, Procps, Psmisc, Sed, Shadow, Tar, Util-linux, Zlib ;
LSB C++:	GCC;
LSB Desktop:	немає;
LSB Runtime Languages:	Perl;
LSB Printing:	немає;
LSB Multimedia:	немає.

Пакети які поставляються BLFS і є необхідними для задоволення вимог LSB

LSB Core:	At, Batch (a part of At), Bc, Cpio, Ed, Fcrontab, Initd-tools, Lsb_release, PAM, Sendmail (or Postfix or Exim);
LSB C++:	немає;
LSB Desktop:	ATK, Cairo, Desktop-file-utils, Freetype, Fontconfig, Glib2, GTK+2, Icon-naming-utils, Libjpeg, Libpng, Libxml2, MesaLib, Pango, Qt3, Qt4, Xorg;
LSB Runtinme Languages:	Python;
LSB Printing:	CUPS;
LSB Multimedia:	Alsa Libraries, NSPR, NSS, OpenSSL, Java, Xdg-utils .

Пакети які не поставляються BLFS і є необхідними для задоволення вимог LSB

LSB Core:	немає;
LSB C++:	немає;
LSB Desktop:	немає
LSB Runtime Languages:	немає;
LSB Printing:	немає;
LSB Multimedia:	немає.

Причина включення пакетів в книзі

Як було установлено раніше, ціль проекту LFS полягає в створенні повної і готової до використання базової системи. Це включає усі пакети, які необхідні для їх копіювання, для забезпечення відносно мінімальну базу, з якої можливо створення більш функціональної системи, базуючись на варіанти вибору користувача. Це не означає, що LFS є найменш можливою системою, яка тільки можлива. Декілька важливих пакетів включено, але вони не є строго обов'язковими. Список, який наведений нижче, документує логічне обґрунтування для кожного пакету в книзі.

- Autoconf.

Цей пакет вміщує в собі програми для створення скриптів командної стрічки, які можуть автоматично конфігурувати вихідний код програми з шаблону розробника. Він часто потрібний для перекомпіляції пакету після оновлень для процесу побудови.

– Automake

Цей пакет має в собі програми для генерації Make-файлів з шаблону. Він часто потрібний для перекомпіляції пакету після оновлень для процесу побудови.

– Bash

Цей пакет необхідний для задоволення стандарту LSB Core і забезпечує інтерфейс Bourne Shell до системи (командну стрічку). Він був вибраний з множини інших пакетів цього інтерфейсу через його широке використання і широкі можливості крім базових функцій командної стрічки.

– Binutils

Цей пакет вміщує версію yacc (Yet Another Compiler Compiler — ще один компілятор компілятора) проекту GNU (www.gnu.org), котрий необхідний для компілювання інших програм LFS.

– Bzip2

Цей пакет має у собі програму для компресії і декомпресії файлів (архіватор). Він розпаковує багато пакетів LFS.

– Check

Цей пакет має у собі тестовий інвентар для інших програм. Він встановлюється як тимчасовий інструмент.

– Coreutils

Цей пакет має в собі число основних програм для маніпуляції файлами і директоріями. Ці програми необхідні для менеджменту файлами у командній стрічці, і також для процедур встановлення кожного пакету в LFS.

– DejaGNU

Цей пакет вміщує в собі програми, котрі показують різницю між файлами і директоріями. Ці програми можуть бути використані для створення патчів і також використовуються у багатьох процедурах побудови (компілювання) програм.

– Expect

Цей пакет має в собі програму для підтримки скриптових діалогів з іншими інтерактивними програмами. Вона поширено використовується для тестування інших програм. Вона встановлюється тільки як тимчасовий інструмент.

– E2fsprogs

Цей пакет постачає утиліти для обробки ext2, ext3 і ext4 файлових систем. Це є найбільш широко розповсюджені і ретельно тестовані файлові системи які підтримує Лінук.

– File

Цей пакет забезпечує вас утилітою для визначення типу даного файлу чи файлів. Деякі пакети потребують

його для компілювання.

– Findutils

Цей пакет має у собі утиліту для генерування програм які розпізнають шаблони у тексті. Це є версія GNU програми lex (lexical analyzer). Вона необхідна для побудови декількох пакетів LFS.

– Gawk

Цей пакет розповсюджує програму для маніпулювання текстовими файлами. Це є версія GNU awk (Aho-Weinberg-Kernighan). Вона використовується в багатьох інших пакетних скриптах побудови.

– Gcc

Цей пакет є колекцією компіляторів GNU (GNU Compiler Collection). Вона вміщає C і C++ компілятори без яких ви не побудуєте нічого.

– GDBM

Цей пакет вміщає бібліотеку Менеджера Баз даних GNU (GNU Database Manager library). Вона використовується іншим LFS пакетом, Man-DB.

– Gettext

Цей пакет вміщає утиліти і бібліотеки для інтернаціоналізації і локалізацій цілого числа пакетів.

– Glibc

Цей пакет вміщає головну бібліотеку C функцій. Програми, які працюють під Лінукс, не будуть функціонувати без неї.

– GMP

Цей пакет вміщає математичну бібліотеку яка поширює корисні функції для довільної арифметичної точності. Вона необхідна для побудови Gcc.

– Grep

Цей пакет вміщає програми для пошуку крізь файли. Ці програми використовуються більшістю скриптами побудови програм.

– Groff

Цей пакет розповсюджує програми для обробки і форматування тексту. Одна важлива функція цих програм є форматування системних сторінок допомоги (man pages).

– GRUB

Цей пакет називається Grand Unified Boot Loader (головний єдиний завантажувач системи). Він є одним з декількох можливих завантажувачів, але він також є найбільш гнучкий.

– Gzip

Цей пакет має у собі програму для компресії та декомпресії файлів. Він необхідний, щоб розархівовувати багато пакетів у LFS і поза ним.

- Iana-etc

Цей пакет забезпечує дані для мережних сервісів і протоколів. Він є необхідним, щоб увімкнути відповідні мережеві можливості.

- Inetutils

Цей пакет має в собі програми для базового адміністрування мереж.

- IProute2

Цей пакет вміщає в собі програми для базової і розширеної підтримки протоколів IPv4 і IPv6. Він був вибраний з багатьох утиліт для мереж (net-tools) через його сумісність з IPv6.

- Kbd

Цей пакет вміщає таблицю кодів, клавіатурні утиліти для не US-клавіатур і певну множину консольних шрифтів.

- Kmod

Цей пакет містить програми, які необхідні для адміністрування модулів ядра.

- Less

Цей пакет забезпечує дуже гарним і зручним переглядачем текстових файлів, дозволяє виконувати переміщення вверх і вниз по текстовому файлу. Він також використовується пакетом Man-DB для переглядання сторінок документації системи (manpages).

- Libtool

Цей пакет вміщує загальний скрипт підтримки бібліотек GNU (generic library support script). Він приховує складність використання бібліотеки спільного користування (shared libraries). Вона необхідна для тестових наборів інших пакетів LFS.

- Linux Kernel (ядро Лінукс)

Цей пакет є ядро усієї Операційної Системи, без якого її робота неможлива.

- M4

Цей пакет вміщує загальний текстовий макро процесор (general text macro processor) — корисний як будівничиа програма для інших програм.

- Make

Цей пакет вміщає вміщує програму для направлення побудови пакетів. Вона вимагається майже усіма пакетами у LFS і поза ним.

- Man-DB

Цей пакет вміщує програми для знаходження і перегляду сторінок документації системи (man pages). Вона вибрана замість пакету man (manual — мануал, підручник) через його чудові інтернаціональні можливолсті.

- Man-pages

Цей пакет вміщує фактичний контент базових сторінок документації Лінукс системи і ОС в загальному.

– MPC

Цей пакет вміщує функції для арифметичних операцій з комплексними числами. Він необхідний для GCC.

– MPFR

Цей пакет вміщує функції для роботи з числами з різною точністю. Від нього залежний GCC пакет.

– Ncurses

Цей пакет розповсюджує бібліотеки для термінально незалежної обробки символів екрану. Вона дуже часто забезпечує контроль курсором для системи меню. Вона потрібна для певного числапакетів.

– Patch

Цей пакет вміщує програми для модифікації чи створення файлів застосовуючи файли-патчі, які зазвичай створені програмою diff. Вона необхідна пдля процедури побудови для багатьох пакетів.

– Perl

Цей пакет є інтерпретатором для скриптової мови PERL. Він необхідний для встановлення і тестових наборів для багатьох пакетів LFS.

– Pkg-config

Цей пакет забезпечує програмою, яка видає мата-дані про встановлені бібліотеки чи пакети.

– Popt

Цей пакет є бібліотекою, яка використовується деякими програмами для розбору введених рядків у командний інтерпретатор.

– Procps

Цей пакет вміщує програми для моніторингу процесів. Ці програми є корисні для системних адміністраторів, і такж використовуються скриптами завантаження LFS.

– Psmisc

Цей пакет вміщує програми для виведення інформації про поточні виконувані процеси в системі. Ця програма є корисною для системних адміністраторів.

– Readline

Цей пакет є набором бібліотек, котрі уможливлюють корегування команд у командному інтерпретаторі і можливість перегляду історії команд. Ця бібліотека використовується програмою Bash.

– Sed

Цей пакет дозволяє редагування тексту без відкривання його в текстових редакторах. Він також необхідний для більшості конфігураційних скриптів пакетів LFS.

– Shadow

Цей пакет вміщує програму для ведення журналу системних повідомлень, такі як повідомлення ядра чи повідомлення процесів демонів (ті, що виконуються в фоновому режимі і не прив'язані до якого-небудь терміналу) коли відбувається незвичайна подія (помилка тощо).

- Sysinit

Цей пакет забезпечує систему програмою init, котра є батьківською для усіх інших процесів у системах Лінукс.

- Tar

Цей пакет забезпечує архівування і розархівування. Її використовують усі пакети LFS.

- Tcl

Цей пакет вміщає програму Tool Command Languag, яка використовується у багатьох наборах тестів пакетів LFS. Вона встановлюється як тимчасовий інструмент.

- Texinfo

Цей пакет вміщує програму для редагування, написання і перетворення інформаційних сторінок. Вона використовується у процедурах встановлення багатьох пакетів LFS.

- Udev

Цей пакет вміщує програму для динамічного створення файлів пристроїв. Це є альтернатива до створення тисячі статичних пристроїв у директорії /dev.

- Util-linux

Цей пакет має в собі різні утиліти. Між ними є програми для обробки файлових систем, консолей, розділів і повідомлень.

- Vim

Цей пакет розповсюджує редактор. Він був вибраний через його сумісність з класичним редактором vi і його велику кількість сильних можливостей. Цей редактор є дуже персональним вибором багатьох користувачів і інший текстовий редактор може замінити його при бажанні.

- XZ Utils

Цей пакет вміщує програми для компресії і декомпресії файлів. Він дає можливість найвищої компресії загально можливої і є корисним для декомпресії пакетів у форматі XZ чи LZMA.

- Zlib

Цей пакет вміщує підпрограми для компресії і декомпресії, які використовуються деякими програмами.

Передумови

Будувати систему LFS не є легким завданням. Воно вимагає певного рівня знань у сфері системного адміністрування у системах Unix (Юнікс), в порядку розв'язання проблем і коректного виконання вказаних команд. Зокрема, як абсолютний мінімум, ви повинні вже мати здатність до використання командної стрічки (shell) для копіювання чи переміщення файлів і директорій, дивитись вміст папок і директорій, і змінювати

поточну директорію. Також очікувано, що ви маєте прийнятні знання у використанні і встановленні програм Лінукс.

Оскільки книга LFS припускає, що ви маєте принаймні цей базовий рівень навичок, деякі форуми підтримки користувачів LFS навряд чи будуть надавати вам допомогу у цих сферах. Вам навряд чи будуть відповідати на питання по такому базовому рівню, або вас будуть направляти до основного списку попередньо читання LFS (списку, який вказує, що необхідно прочитати перед тим як читати і діяти за інструкціями цієї книги).

Перед тим як будувати систему LFS, ми рекомендуємо прочитати наступні сторінки документації (англ. HOWTOs — як зробити...):

- Як побудувати (скомпілювати) програмний пакет (Software-Building-HOWTO) *http://www.tldp.org/HOWTO/Software-Building-HOWTO.html* (англ.)

Ця інтернет сторінка є зрозумілий гід (путівник тощо) по шляхам побудови (компілювання) і встановлення програм у “загальних” пакетах Unix (Юнікс) під Лінуксом. Хоча він був написаний деякий час назад, він досі забезпечує хорошу суму базових технік, які необхідні для побудови і встановлення програмного забезпечення.

- Користувацький гід по системі Лінукс (The Linux Users' Guide) *http://www.linuxhq.com/guides/LUG/guide.html*

Цей гід охоплює використання відбірного програмного забезпечення у Лінукс. Ця довідка є також дуже старою.

- Основна підказка для попереднього прочитання (The Essential Pre-Reading Hint) *http://www.linuxfromscratch.org/hints/downloads/files/essential_prereading.txt*

Це є підказка написана від проекту LFS спеціально для користувачів які є новачками в Лінукс. Вона включає список посилань на чудові джерела інформації на широкий діапазон тем. Будь-який користувач, який намагається встановити LFS систему повинен мати розуміння багатьох тем у цій підказці.

Вимоги до комп'ютера

Твоя хост-система (комп'ютер) повинна мати наступне програмне забезпечення з мінімально вказаними номерами версій. Це не повинно бути проблемою для більшості сучасних дистрибутивів Лінукс. Також майте на увазі, що багато дистрибутивів помістить заголовки пакетів (файли які підключаються до коду з метою забезпечення програм відповідними функціями) в окремі пакети, часто у формі “<назва-пакету>-devel” або “<назва-пакету>-dev” (“<package-name>-devel”, “<package-name>-dev”). Будьте впевнені що ви їх встановили, якщо ваш дистрибутив постачає їх.

Старіші версії програмного забезпечення у наступному списку можуть працювати, але не були тестовані.

- **Bash-3.2** (/bin/bash повинен бути символічним або жорстким посиланням на програму bash);
- **Binutils-2.17** (Версії вищі ніж 2.22 не рекомендовані, так як не були ще тестовані);
- **Bison-2.3** (/usr/bin/yacc повинен бути посиланням на програму bison або малим скриптом, який його її запускає);
- **Bzip2-1.0.4**;
- **Coreutils-6.9**;

- **Diffutils-2.8.1**;
- **Findutils-4.2.31**;
- **Gawk-3.1.5** (/usr/bin/awk повинен бути посиланням на gawk);
- **Gcc-4.1.2** (версії вищі ніж 4.7.1 не рекомендуються так як не були ще тестовані);
- **Grep-2.5.1a**;
- **Gzip-1.3.12**;
- **Linux Kernel-2.6.25** (яке було скомпільоване Gcc версією 4.1.2 або старшою).

Причина вимоги такої версії ядра є тому, що ми вказали цю версію коли будували glibc у Розділі 6 як рекомендацію розробників. Ця версія також є вимогою для udev.

Якщо ядро хост-системи є раніше ніж 2.6.25, або воно не було скомпільоване використовуючи компілятор GCC-4.1.2 (або пізнішої версії), ви будете змушені замінити ядро дотримуючись специфікацій. Є два шляхи як ви можете поступити з цим. Спочатку, погляньте чи ваш постачальник дозволяє встановлювати ядро версії 2.6.25 або пізнішої. Якщо так, ви можете встановити його. Якщо ж вам не постачається пакет ядра прийнятної версії, або ви не хотіли б встановлювати його, ви можете скомпілювати його самостійно. Інструкції для компіляції ядра і конфігурування системного завантажувача (припустимо, що це GRUB) розміщено в Розділі 8.

- **M4-1.4.10**;
- **Make-3.81**;
- **Patch-2.5.4**;
- **Perl-5.8.8**;
- **Sed-4.1.5**;
- **Tar-1.18**;
- **Texinfo-4.9**;
- **Xz-5.0.0**.

Зверніть увагу, що посилання згадані вище вимагаються, щоб побудувати систему LFS використовуючи інструкції які розміщені всередині цієї книги. Посилання які вказують на інше програмне забезпечення (такі як dash, mawk тощо) можуть працювати, але не протестованими чи підтримувані командою розробників LFS, і можуть вимагати або відхилення від інструкцій або додаткові патчі до деяких пакетів.

Щоб побачити чи ваша хост-система має усі відповідні версії програм і здатність компілювання, виконайте наступне:

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

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

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

gcc --version | head -n1
ldd --version | head -n1 | cut -d" " -f2- # glibc version
grep --version | head -n1
gzip --version | head -n1
cat /proc/version
m4 --version | head -n1
make --version | head -n1
patch --version | head -n1
echo Perl `perl -V:version`
sed --version | head -n1
tar --version | head -n1
echo "Texinfo: `makeinfo --version | head -n1`"
xz --version | head -n1

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

bash version-check.sh
```

Форматування

Для того щоб полегшити читання книги, є кілька конвенцій використаних в цій книзі. Ця секція вміщає деякі приклади форматування.

```
./configure —prefix=/usr
```

Ця форма тексту є розроблена для того, щоб бути набраною так само як ми її бачимо, якщо тільки по-іншому

не вказано у навколишньому тексті. Він також використовується у роз'ясненні секцій щоб визначити, яка з команд на яку посилається.

У деяких випадках, логічний рядок є розбитий на два або більше рядків з зворотнім слешом (backslash - \) в кінці рядка.

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

Зверніть увагу на те, що зворотний слеш повинен бути перед зворотом каретки (введеним новим рядком). Інші пробільні символи, такі як пробіли або табуляція будуть створювати неправильні результати.

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

Ця форма тексту (текст фіксованої довжини) показує вміст екрану, зазвичай як результат виконання команд. Цей формат також використовується для того, щоб показувати назви файлів, такі як /etc/ld.so.conf.

Наголошення

Ця форма тексту використовується для декількох цілей в книзі. Його головна ціль — наголосити на важливі частини пунктів.

http://www.linuxfromscratch.org/

Цей формат використовується для гіперпосилань до сторінок LFS общини і до зовнішніх сторінок. Воно включає сторінки документації (HOWTOs — як зробити...), сторінки документації і веб-сайти.

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

Цей формат використовується при створенні конфігураційних файлів. Перша команда говорить системі створити файл `$LFS/etc/group` з вмістом який набраний на клавіатурі у вигляді наступних рядків, фж поки не зустрінеться послідовність End Of File (EOF).Отже, ця секція в загальному набирається як ми її бачимо.

<ТЕКСТ ДЛЯ ЗАМІНИ>

Цей формат використовується для заміни тексту який не є для набирання так, як ми його бачимо або для операцій копіювання-вставки.

[ОПЦІОНАЛЬНИЙ ТЕКСТ]

Цей формат використовується для вставки тексту, який є опціональним (необов'язковим).

`passwd(5)`

Цей формат використовується для звернення до спеціальної сторінки документації. Номер в середині дужок вказує на спеціальну секцію всередині документації. Для прикладу, **passwd** має дві сторінки документації. В документації про встановлення LFS, ці дві сторінки документації будуть розміщені в `/usr/share/man/man1/passwd.1` і `/usr/share/man/man5/passwd.5` . Коли книга використовує passwd(5) вона спеціально посилається до `/usr/share/man/man5/passwd.5`. **man passwd** виведе першу сторінку в якій знайде збіг “passwd”, якою є сторінка /usr/share/man/man1/passwd .1. Для цього прикладу, вам необхідно виконати **man 5 passwd,** для прочитання специфічної сторінки, яку вам рекомендували. Ви повинні запам'ятати що більшість сторінок документації не мають дубльовані назви у різних секціях. Таким чином, **man <program name>** зазвичай достатньо.

Структура

Дана книга поділена на наступні частини.

Частина I — Вступ

Частина I роз'яснює кілька важливих пунктів про те, як триває інсталювання LFS. Ця секція також забезпечує мета-інформацією про дану книгу.

Частина II — Підготовка до побудови

Частина II описує необхідні підготування для процесу побудови: створення розділів, завантажування пакетів і компілювання тимчасових інструментів.

Частина III — Побудова системи LFS

Ця частина проводить читача через процес побудови системи LFS — компілювання і встановлення усіх пакетів один за одним, встановлення скриптів завантаження і встановлення ядра ОС. Кінцева система Лінукс є базою на якій інше програмне забезпечення, за бажанням, може бути побудоване для розширення її можливостей. В кінці цієї книги розміщений легкий для користування список, який вкаже вам на встановлені програми, бібліотеки і важливі файли.

Помилки

Програмне забезпечення, яке використовувалося для створення LFS системи, постійно оновлюється і розширюється. Попередження до безпеки і виправлення програмних помилок може ставати доступним після того, як книга LFS була випущена. Для того, щоб перевірити чи версії пакетів або інструкції у цьому випуску LFS потребують яких-небудь модифікацій для виправлення вразливостей безпеки чи виправлення помилок програм, будь ласка, відвідайте сторінку *http://www.linuxfromscratch.org/lfs/errata/7.2/* перед тим, як розпочати ваш процес побудови. Ви повинні запам'ятати будь-які знайдені зміни і приймати їх до відповідних секцій книги під час того, як ви будете будувати свою власну систему LFS.

Частина I. Вступ

Глава 1. Вступ

1.1. Як побудувати LFS систему

Система LFS буде побудована використовуючи вже встановлену систему Лінукс дистрибутива (такі як Debian, Mandriva, Red Hat або SUSE). Встановлена система (хост) буде використана як відправна точка для забезпечення необхідними програмами, включаючи компілятор, компонувальник, і командний інтерфейс (shell) для того, щоб побудувати на цій базі нову систему. Виберіть опцію “розробник” (“development”) під час встановлення дистрибутива, щоб мати можливість доступу до відповідних інструментів.

Як альтернатива до встановлення окремого дистрибутиву на вашу машину, ви можете використовувати LiveCD (завантажувальний диск) з комерційного дистрибутиву.

Частина 2 цієї книги описує як створити новий “рідний” для Лінукс розділ з відповідною файловою системою. Це є місце де нова система LFS буде скомпільована і встановлена. Частина 3 роз'яснює які пакети і патчі необхідні для завантаження для того щоб побудувати систему LFS і як зберегти їх на новій файловій системі. У частині 4 обговорюється встановлення відповідного робочого середовища (working environment). Будь-ласка, прочитайте Частину 4 уважно, так як вона роз'яснює деякі важливі питання, які вказують що потрібно знати перед тим як почати йти вашим вибраним шляхом крізь Частину 5 і наступних.

Частина 5 також демонструє вам як побудувати перший рубіж (етап) інструментів, включаючи Binutils і GCC (перший рубіж означає, що ці два основні пакети будуть перевстановлені). Наступним кроком буде збирання Glibc — бібліотеку мови програмування C. Glibc буде скомпільована програмними інструментами, які були побудовані у першому рубежі. Після цього, буде наступний етап, в якому ви побудуєте другий набір інструментів. Він використовуватиме динамічні бібліотеки. Решта пакетів з Частини 5 будуть збудовані використовуючи інструменти з другого набору. Коли це все буде зроблено, процес інсталяції LFS більше не залежатиме від іншого дистрибутиву, за винятком робочого ядра.

Ці зусилля щоб ізолювати новозбудовану систему від стороннього дистрибутиву можу здатись надмірною. Повне технічне роз'яснення чому це все зроблено таким шляхом розміщене у Секції 5.2 “Технічні замітки програмного інструментарію”.

Коли ви доберетесь до Частини 6, уся система буде збудована. Програма **chroot** (change root — змінити корінь) застосовується для використання віртуального середовища і відкриває нову сесію командного інтерпретатора, коренева директорія якого знаходиться у розділі, який виділений під систему LFS. Система не перезавантажується, а змінює кореневу директорію, це робиться для того, щоб створити систему, яка здатна до завантаження, необхідно провести деякі додаткові роботи, які в даний момент не потрібні. Основна перевага “chroot-інгу” (зміни кореневого каталогу) в тому, що це дозволяє вам продовжувати використовувати хостову систему, поки будується система LFS. Чекаючи закінчення компіляції пакету, ви можете продовжувати використовувати ваш комп'ютер як зазвичай.

Для закінчення встановлення, скрипти завантаження LFS (LFS-Bootscripts) встановлюються в Частині 7, а ядро і завантажувач системи встановлюються у Частині 8. Частина 9 вміщує інформацію як продовжувати працювати з LFS поза межами даної книги. Після виконання усіх кроків даних у книзі, комп'ютер буде готовий до перезавантаження в нову систему LFS.

Цей процес відбувається в командній стрічці. Детальну інформацію по кожному кроці ви можете отримати у наступних главах і документації яка поставляється з пакетами. Пункти, які можуть здаватись складними, будуть уточнюватись, і все буде ставати на своє місце коли ви розпочнете пригоду, яка називається LFS.

1.2. Що нового з'явилося з останнього випуску

Нижче розміщений список оновлених пакетів зроблених після останнього випуску книги.

Оновлено до:

- Autoconf 2.69 ;
- Automake 1.12.3 ;
- Bison 2.6.2 ;
- Coreutils 8.19 ;
- E2fsprogs 1.42.5 ;
- File 5.11 ;
- Flex 2.5.37 ;
- Gawk 4.0.1 ;
- GCC 4.7.1 ;
- Glibc 2.16.0 ;
- GMP 5.0.5 ;
- Grep 2.14 ;
- Gzip 1.5 ;
- IPRoute2 3.5.1 ;
- Kbd 1.15.3 ;
- Kmod 9 ;
- Libpipeline 1.2.1 ;
- Linux 3.5.2 ;-
- Man-DB 2.6.2 ;
- Man-pages 3.42 ;
- MPC 1.0 ;
- MPFR 3.1.1 ;
- Perl 5.16.1 ;
- Psmisc 22.19 ;
- Shadow 4.1.5.1 ;
- TCL 8.5.12 ;
- Udev 188 (взятий з systemd-188) ;

- Util-Linux 2.21.2 .

Додано:

- bash-4.2-fixes-8.patch ;
- binutils-2.22-build_fix-1.patch ;
- coreutils-8.19-i18n-1.patch ;
- flex-2.5.37-bison-2.6.1-1.patch ;
- glibc-2.16.0-res_query_fix-1.patch ;
- kbd-1.15.3-upstream_fixes-1.patch ;
- make-3.82-upstream_fixes-2.patch ;
- perl-5.16.1-libc-2.patch ;
- pkg-config-0.27 ;
- sed-4.2.1-testsuite_fixes-1.patch ;
- tzdata 2012e .

Видалено:

- bash-4.2-fixes-4.patch ;
- coreutils-8.15-i18n-1.patch ;
- coreutils-8.15-uname-1.patch ;
- flex-2.5.35-gcc44-1.patch ;
- gcc-4.6.2-cross_compile-1.patch ;
- gcc-4.6.2-startfiles_fix-1.patch ;
- glibc-2.14.1-fixes-1.patch ;
- glibc-2.14.1-gcc_fix-1.patch ;
- glibc-2.14.1-cpuid-1.patch ;
- glibc-2.14.1-sort-1.patch ;
- mpfr-3.1.0-fixes-1.patch ;
- perl-5.14.2-libc-1.patch ;
- perl-5.14.2-security-1.patch ;
- shadow-4.1.5-nscd-1.patch .

1.3. Журнал змін

Це є книга Лінукс З Початків версії 7.2, датована 1 вереснем 2012 року. Якщо цій книзі більше ніж пів року, нова і краща версія напевно вже доступна. Щоб це вияснити, будь-ласка перевірте один з наших ftp-зеркал

список яких розміщений на сторінці *linuxfromscratch.org/mirrors.html*.

Нижче розміщений список змін зроблених від останнього випуску книги.

Записи журналу:

- 2012-09-01
  - [bdubbs] - LFS-7.2 released.
- 2012-08-31
  - [bdubbs] - Fix spelling typos in bootscripts.
- 2012-08-29
  - [bdubbs] - Fix spelling typos. Thanks to Gilles Espinasse.
  - [bdubbs] - Add additional explanations for time zone installation, udev, and network configuration.
- 2012-08-27
  - [bdubbs] - Add patch to fix glibc occasional crash with problem nameservers. Fixes #3172.
  - [bdubbs] - Add instructions to Chapter 5 glibc to add rpc headers to the host system if they are missing.
- 2012-08-26
  - [bdubbs] - Install both .tab files in tzdata.
  - [bdubbs] -he host system if they are missing.
- 2012-08-26
  - [bdubbs] - Install both .tab files in tzdata.
  - [bdubbs] - Apply upstream patches to make.
- 2012-08-24
  - [ken] - Remove redundant sed from automake.
- 2012-08-22
  - [bdubbs] - Update glibc text removing noatime mount caution and text regarding test issues.
  - [bdubbs] - Fix packaging for udev-lfs tarball.
- 2012-08-21
  - [bdubbs] - Update udev-lfs tarball for BLFS compatibility.
  - [ken] - Add four locales to the minimum set for test coverage.
  - [ken] - Add patch to fix sed utf8 regression test failures.
  - [bdubbs] - Update statistics for packages.
- 2012-08-20

- [bdubbs] - Upgrade to coreutils-8.19. Fixes #3163.
- [bdubbs] - Upgrade to grep-2.14. Fixes #3164.
- [ken] - Fix how the timezones are installed.
- [bdubbs] - Add patch to fix Flex regression test failures.

- 2012-08-15
  - [bdubbs] - Upgrade to linux-3.5.1. Fixes #3154.
  - [bdubbs] - Upgrade to man-pages-3.42. Fixes #3159.
  - [bdubbs] - Upgrade to automake-1.12.3. Fixes #3161.
  - [bdubbs] - Move shadow to before coreutils to have su available.
- 2012-08-15
  - [bdubbs] - Upgrade to coreutils-8.18. Fixes #3157.
- 2012-08-14
  - [bdubbs] - Upgrade to perl-5.16.1. Fixes #3155.
  - [bdubbs] - Removed unneeded sed instruction from Chapter 6 perl. Fixes #3160.
- 2012-08-13
  - [bdubbs] - Upgrade to flex-2.5.37. Fixes #3139.
  - [matthew] - Upgrade to IPRoute2-3.5.1. Fixes #3158.
- 2012-08-12
  - [bdubbs] - Update to tzcode2012e. Fixes #3156.
  - [bdubbs] - Update to udev (systemd)-188. Fixes #3152.
- 2012-08-06
  - [matthew] - Install a couple more files from the tzdata tarball so that tzselect works again.
- 2012-08-05
  - [matthew] - Upgrade to IPRoute2-3.5.0. Fixes #3148.
  - [matthew] - Upgrade to Tcl-8.5.12. Fixes #3147.
  - [matthew] - Upgrade to E2fsprogs-1.42.5. Fixes #3146.
  - [matthew] - Upgrade to MPC-1.0. Fixes #3142.
  - [matthew] - Upgrade to Bison-2.6.2. Fixes #3140.
  - [matthew] - Upgrade to Linux-3.5. Fixes #3138.
  - [matthew] - Upgrade to Glibc-2.16.0. Fixes #3131.

- 2012-07-25
  - [bdubbs] - Minor fixes to udev-lfs tarball.
- 2012-07-22
  - [bdubbs] - Update to udev (systemd)-187. Fixes #3143.
  - [bdubbs] - Fix udev-retry boot script for latest udev functioanlity. Remove 'udev info --run-dir'
  - [bdubbs] - Update to pkg-config-0.27. Remove popt. Fixes #3141.
- 2012-07-19
  - [bdubbs] - Added Time Zone data package to packages section of the book.
  - [bdubbs] - Added Check to the Rationale section of the book.
- 2012-07-17
  - [matthew] - Upgrade to Linux-3.4.5. Fixes #3137.
- 2012-07-16
  - [bdubbs] - Updated udev to version 186. This update has a major procedure change due to merging systemd
  - and udev. Fixes #3098.
- 2012-07-14
  - [matthew] - Correct the fix for Automake's testsuite. Thanks to Fernando de Oliveira for the report.
- 2012-07-13
  - [matthew] - Apply latest upstream patches for Bash. Fixes #3135.
  - [matthew] - Upgrade to Automake-1.12.2. Fixes #3134.
  - [matthew] - Upgrade to MPFR-3.1.1. Fixes #3133.
  - [matthew] - Upgrade to Grep-2.13. Fixes #3132.
  - [matthew] - Fix Kmod's test suite on x86 hosts. Fixes #3129.
  - [matthew] - Upgrade to Psmisc-22.19. Fixes #3127.
  - [matthew] - Upgrade to Linux-3.4.4. Fixes #3126.
- 2012-07-11
  - [bdubbs] - Update to GRUB-2.00. Fixes #3130.
- 2012-07-10
  - [bdubbs] - Specify PKG_CONFIG_PATH for libpipeline checks. Fixes #3120.
- 2012-06-23
  - [matthew] - Upgrade to XZ-5.0.4. Fixes #3125.

- [matthew] - Upgrade to Kmod-9. Fixes #3124.
- [matthew] - Upgrade to Psmisc-22.18. Fixes #3123.
- [matthew] - Upgrade to Man-DB-2.6.2. Fixes #3122.
- [matthew] - Upgrade to Gzip-1.5. Fixes #3121.
- [matthew] - Upgrade to GCC-4.7.1. Fixes #3117.
- [matthew] - Upgrade to E2fsprogs-1.42.4. Fixes #3116.
- [matthew] - Upgrade to Linux-3.4.3. Fixes #3114.

- 2012-06-17
  - [bdubbs] - Fix install error in iproute2. Fixes #3119.
  - [bdubbs] - Update rare issues in bootscripts when using LVM or initramfs.
  - [bdubbs] - Add note about automake run time for tests. Fixes #3118.
- 2012-06-10
  - [ken] - kbd-1.15.3 : go back to changing configure, and touch aclocal.m4 : thanks to Bryan for explaining the
  - problem.
- 2012-06-07
  - [matthew] - Remove --disable-perl-regexp switch from chapter 5's Grep instruction. It should be unnecessary
  - now as there should be no way for the host's libraries to leak through to the chapter 5 toolchain. Thanks to
  - Jeremy Huntwork for the report.
- 2012-06-06
  - [matthew] - Upgrade to Bison-2.5.1. Fixes #3112.
- 2012-06-05
  - [matthew] - Remove a couple of sed commands from Binutils' instructions, as the tests have been fixed
  - upstream. Thanks to Waleed Hamra for the report.
  - [matthew] - Upgrade to Linux-3.4.1. Fixes #3110.
  - [ken] - Really remove the redundant program resizecons from kbd, by changing configure.ac instead of
  - configure. Thanks to xinglp.
- 2012-06-04
  - [bdubbs] - Incorporate perl fixes from 2012-06-03 in the perl patch.
  - [matthew] - Upgrade to Psmisc-22.17. Fixes #3109.

- [matthew] - Upgrade to Automake-1.12.1. Fixes #3106.
- [matthew] - Apply latest upstream patches for Bash. Fixes #3103.

- 2012-06-03
  - [bdubbs] - Add pkg-config-0.26-internal-glib to the book. Fixes #3105.
  - [bdubbs] - Add popt-1.16 to the book.
  - [bdubbs] - Update Chapter 5 perl instructions for the LFS environment. Fixes #3104.
- 2012-05-30
  - [bdubbs] - Copy all entries in /lib/udev/devices to /dev in mountvirtfs. Fixes #3102.
  - [matthew] - Correct the location of various package's man pages. Fixes #3097.
  - [matthew] - Upgrade to Util-Linux-2.21.2. Fixes #3100.
  - [matthew] - Upgrade to Perl-5.16.0. Fixes #3094.
  - [matthew] - Upgrade to IPRoute2-3.4.0. Fixes #3096.
  - [matthew] - Upgrade to Linux-3.4. Fixes #3092.
  - [matthew] - Upgrade to E2fsprogs-1.42.3. Fixes #3091.
- 2012-05-24
  - [bdubbs] - Minor tweaks to mountkernfs boot script. Also fixes #3093.
- 2012-05-20
  - [bdubbs] - Remove a bashism from the mountkernfs boot script.
  - [bdubbs] - Move the $time init capability from setclock to udev. Fixes #3085.
  - [bdubbs] - Remove and recreate $LFS/dev/shm in Section 6.2 if it is a symbolic link. Fixes #3085.
- 2012-05-18
  - [ken] - Remove the redundant program resizecons from kbd and remove its man page (program was only
  - installed on i?86, but man page was always installed).
- 2012-05-14
  - [matthew] - Upgrade to Linux-3.3.6. Fixes #3089.
- 2012-05-13
  - [matthew] - Upgrade to Coreutils-8.17. Fixes #3083.
- 2012-05-11
  - [matthew] - Upgrade to Man-Pages-3.41. Fixes #3084.
  - [matthew] - Upgrade to Linux-3.3.5. Fixes #3080.

- [matthew] - Upgrade to GMP-5.0.5. Fixes #3079.
- [matthew] - Remove sed from GCC pass 2 and chapter 6, which prevented the fixincludes script from being
- run; it is no longer run by default. Reported by Jeremy Huntwork.

- 2012-05-10
  - [bdubbs] - Add /etc/lsb-release file in Chapter 9.
- 2012-05-09
  - [bdubbs] - Update LSB packages in BLFS.
- 2012-05-06
  - [matthew] - Upgrade to Zlib-1.2.7. Fixes #3078.
  - [matthew] - Apply latest upstream patches for Bash. Fixes #3077.
  - [matthew] - Add back a patch for Glibc that prevents various BLFS programs, such as aplay, from segfaulting.
- 2012-05-05
  - [bdubbs] - Various minor text changes to both book and bootscripts.
- 2012-05-01
  - [ken] - Upgrade to Kbd-1.15.3. Fixes #2990.
- 2012-04-29
  - [matthew] - Upgrade to Linux-3.3.4. Fixes #3074.
  - [matthew] - Upgrade to Man-Pages-3.40. Fixes #3072.
  - [matthew] - Upgrade to Autoconf-2.69. Fixes #3071.
  - [matthew] - Upgrade to Automake-1.12. Fixes #3070.
  - [matthew] - Upgrade to Grep-2.12. Fixes #3068.
  - 2012-04-26
  - [ken] - tidy some minor issues from the merge.
- 2012-04-25
  - [bdubbs] - Incorporate changes developed and tested in the jh branch.
  - [jhuntwork] - Update chapter 5 toolchain technical notes to match changes in build method.
  - [jhuntwork] - Use --with-native-system-header-dir switch in chapter 5 gcc. This replaces seds that were used
  - previously to alter the CROSS_SYSTEM_HEADER_DIR and NATIVE_SYSTEM_HEADER_DIR values to

- keep the toolchain searching for headers only in /tools/include and not /usr/include. Thanks to Pierre Labastie.
- Fixes #3066.
- [jhuntwork] - Remove --without-cloog and --without-ppl from chapter 5 gcc. These are unnecessary since it
- doesn't matter if pass 1 gcc is linked against host libs and it should be impossible for the build of pass 2 gcc to
- find host headers or libs.
- [jhuntwork] - Adjust build method to use sysroot.

- 2012-04-24
  - [matthew] - Upgrade to Linux-3.3.3. Fixes #3067.
  - [matthew] - Upgrade to Man-Pages-3.39. Fixes #3065.
  - [matthew] - Upgrade to Kmod-8. Fixes #3064.
- 2012-04-19
  - [bdubbs] - Change two group IDs to support a legacy program. Fixes #3061.
- 2012-04-15
  - [matthew] - Upgrade to Linux-3.3.2. Fixes #3063.
  - [matthew] - Upgrade to Automake-1.11.5. Fixes #3062.
  - [matthew] - Use su from chapter 6 Coreutils in the Bash instructions, instead of the one from chapter 5. Install
  - su as su rather than su-tools in chapter 5. Fixes #3057.
- 2012-04-09
  - [bdubbs] - Update networking bootscripts. See bootscripts change log for details. Fixes #3053.
- 2012-04-05
  - [bdubbs] - Change the location for the python gdb module generated by gcc to the correct location. Fixes
  - (again) #3048.
- 2012-04-03
  - [matthew] - Upgrade to Linux-3.3.1. Fixes #3059.
  - [matthew] - Upgrade to Automake-1.11.4. Fixes #3058.
  - [matthew] - Upgrade to Gawk-4.0.1. Fixes #3056.
  - [matthew] - Upgrade to Util-Linux-2.21.1. Fixes #3055.

- [matthew] - Upgrade to E2fsprogs-1.42.2. Fixes #3051.
- [matthew] - Upgrade to Coreutils-8.16 and drop the uname patch. Fixes #3048.

- 2012-03-28
  - [bdubbs] - Move a python module for gdb generated by gcc to a better location. Fixes #3048.
  - [bdubbs] - Adjust minimum version of xz-utils in Host Requirements.
  - [bdubbs] - Reword description of log files in section Creating Essential Files.
- 2012-03-27
  - [matthew] - Add a patch to fix building of Binutils with the -O3 compiler flag. Thanks to Pierre Labastie for
  - the report.
  - [matthew] - Add the GCC fix patch back to Glibc instructions to fix a build issue on 32-bit hosts. Thanks to
  - Pierre Labastie for the report.
- 2012-03-26
  - [matthew] - Upgrade to Man-Pages-3.38. Fixes #3047.
  - [matthew] - Upgrade to E2fsprogs-1.42.1. Fixes #3046.
  - [matthew] - Upgrade to Glibc-2.15. Fixes #3045. Thanks to Andy Benton for the patch.
  - [matthew] - Upgrade to GCC-4.7.0. Fixes #3044. Thanks to Andy Benton for the patch.
  - [matthew] - Upgrade to IPRoute2-3.3.0. Fixes #3043.
- 2012-03-22
  - [bdubbs] - Fix corner case in ipv4-static script.
- 2012-03-20
  - [matthew] - Upgrade to Linux-3.3. Fixes #3042.
  - [matthew] - Upgrade to Kmod-7. Fixes #3041.
  - [matthew] - Upgrade to Udev-182. Fixes #3040.
- 2012-03-19
  - [bdubbs] - Move optional LVM initialization to the end of the udev boot script so an LVM partition can be
  - used for swap.
- 2012-03-14
  - [matthew] - Remove GCC's cross-compile patch as it isn't required.
  - [matthew] - Apply new upstream patches for Bash. Fixes #3037.

- [matthew] - Upgrade to Linux-3.2.11. Fixes #3036.
- [matthew] - Upgrade to Man-Pages-3.37. Fixes #3034.

- 2012-03-11
  - [matthew] - Workaround an issue in Gettext's configure script that can cause it to hang on certain hosts when
  - determining the path for Emacs Lisp files on certain hosts. Reported by and fix provided by DJ Lucas.
- 2012-03-06
  - [matthew] - Upgrade to Libpipeline-1.2.1. Fixes #3031.
  - [matthew] - Upgrade to Kmod-6. Fixes #3030.
  - [matthew] - Upgrade to Grep-2.11. Fixes #3029.
  - [matthew] - Upgrade to GCC-4.6.3. Fixes #3028.
  - [matthew] - Upgrade to Psmisc-22.16. Fixes #3026.
  - [matthew] - Upgrade to File-5.11. Fixes #3024.
  - [matthew] - Upgrade to Linux-3.2.9. Fixes #3023.
  - [matthew] - Upgrade to Util-Linux-2.21. Fixes #3002.
- 2012-03-02 ;
  - [bdubbs] - LFS-7.1 released.

1.4. Ресурси

1.4.1 FAQ (питання які часто задаються)

Якщо під час побудови LFS системи ви отримаєте якусь помилку, будете мати якесь запитання або вважаєте, що є якась помилка в книзі, будь-ласка, почніть з прочитання Часто Задаваних Питань (Frequently Asked Question — FAQ), яка розміщена за адресою *http://www.linuxfromscratch.org/faq/*.

1.4.2. Список розсилки

Сервер `linuxfromscratch.org` тримає у собі число електронних адрес, які використовуються у розробці проекту LFS. Цей список включає головні списки розсилки розробників і підтримки з поміж-інших. Якщо FAQ не дає відповіді на ваше питання, наступним кроком буде пошук у списку розсилки за адресою *http://www.linuxfromscratch.org/search.html.*

Для отримання інформації з інших списків, як підписатись, адреси архівів і додаткову інформацію, відвідайте *http://www.linuxfromscratch.org/mail.html.*

1.4.3. IRC

Деякі члени товариства LFS пропонують допомогу у нашому товаристві в мережі Internet Relay Chat (IRC). Перед тим, як використовувати цю підтримку, будь-ліска, впевніться у тому, що на ваше запитання ще не

дали відповідь у FAQ-сторінках чи на розсилці товариства LFS. Ви можете знайти IRC-мережу за адресою irc.linuxfromscratch.org . Канал підтримки називається #LFS-support.

1.4.4. Сайти дзеркал

Проект LFS має певну кількість дзеркал у поширених світі, щоб зробити доступ до веб-сайту і завантаження необхідних пакетів більш зручним. Будь-ласка, відвідайте веб-сайт, який розміщений за сторінкою http://www.linuxfromscratch.org/mirrors.html , для того щоб подивитись на список дзеркал.

1.4.5. Інформація для зв'язку

Будь-ласка, направляйте усі ваші запитання і коментарі до однієї з списків розсилки LFS (дивіться вище).

1.5. Допомога.

Якщо проблема чи питання зустрілися під час роботи з цією книгою, будь-ласка, перегляньте FAQ-сторінки за адресою http://www.linuxfromscratch.org/faq/#generalfaq . Відповіді на запитання часто вже є на цих сторінках. Якщо на ваше запитання немає відповіді на цих сторінках, спробуйте знайти джерело помилки. Наступна підказка дасть вам деяке уявлення до пошуку і усунення несправностей: http://www.linuxfromscratch.org/hints/downloads/files/errors.txt.

Якщо ви не можете знайти проблему у списку FAQ, шукайте у розсилці за адресою http://www.linuxfromscratch.org/search. html.

Також, ми маємо чудове товариство LFS, яке готове дати вам пораду по питанням спискам розсилки і IRC (подивіться Частину 1.4 “Ресурси”). Однак, ми отримуємо питання по підтримці кожен день, і на багато з цих питань можна отримати відповідь прочитавши FAQ і зробивши пошук у спискам розсилки. Отож, для того щоб ми надавали найкращу підтримку, яка тільки можлива, ви повинні зробити деяку розвідку самостійно. Це дозволяє нам зробити фокус на більш незвичайних питаннях. Якщо ваші пошуки не дали розв'язку проблеми, будь-ласка, уся відповідна інформація (згадана вище) для вашого прохання про допомогу.

1.5.1. Дещо необхідне до згадування

Крім зрозумілого роз'яснення проблеми яку ми маємо, для отримання допомоги істотно вказати також:

- Версія книги яка використовується (у нашому випадку 7.2);
- Дистрибутив і його версія;
- Вивід скрипту з секції “Вимоги до хосстової системи”;
- Пакет, або секція в якій є проблема;
- Отримане повідомлення про помилку, або симптом.
- Зазначте, де ви відхилились від інструкцій у книзі.

Увага

Відхилення від інструкцій в книзі не означає, що ми вам не допоможемо. В кінці кінців, проект LFS є про персональні вподобання. Будучи проінформованими про які-небудь зміни внесені до встановлених процедур, допомагає нам оцінювати і визначати можливі причини ваших проблем.

1.5.2. Проблеми з скриптами configure (конфігурації)

Якщо щось пішло не так під час виконання скрипту configure, перегляньте файл config.log. Цей файл може містити помилки які надійшли під час конфігурації і які не виводились на екран. Включайте відповідні рядки якщо ви хочете задати питання.

1.5.3. Помилки компіляції

Вивід екрану і вміст деяких файлів є корисним для визначення причини помилок компіляції. Вивід на екран з скрипту configure і команди make можуть бути корисними. Це не є необхідним включати увесь вивід, але робіть включення достатньої кількості інформації. Нижче наведений приклад типу інформації яка необхідна для включення з виводу екрану при виконанні команди make.

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

У цьому випадку, багато людей просто надсилає нам верхню частину помилки

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

Цієї інформації не достатньо, щоб правильно діагностувати проблему тому, що вона нотує тільки те, що щось пішло неправильно, а не що пішло неправильно. Текст, як у прикладі наведеному вище, є тим що необхідно додати до вашого повідомлення тому, що в ньому вказано, яка команда була виконана і відповідний текст помилки(помилок).

Ідеальна стаття про те як правильно просити про допомогу в мережі Інтернеті є доступною за адресою http://catb.org/~esr/faqs/smart-questions.html. Прочитайте і дійте відповідно до інструкцій в даному документі, щоб підвищити ймовірність отримання допомоги, яка вам необхідна.

Частина II. Підготовка до побудови.

Глава 2. Підготовлюємо новий розділ

2.1. Вступ

У цій частині, буде підготовлюватися новий розділ, де буде міститись нова система LFS. Ми створимо самий розділ, файлову систему на ньому, і підключимо (монтуємо — англ. mount) її.

2.2. Створення нового розділу

Як і усі інші операційні системи, LFS встановлюється на спеціально виділений розділ жорсткого диску. Рекомендований підхід до побудови системи LFS — використання доступного пустого розділу жорсткого диску або, якщо ви маєте достатньо місця, створити такий.

Мінімальні системні вимоги — це розділом з принаймні 2.8 гігабайти (Гбайти) простору. Цього буде достатньо для зберігання усіх архівованих пакетів і скомпільованих програм. Однак, якщо система LFS призначена для того щоб бути основною системою Linux, додаткове програмне забезпечення буде встановленим, що в свою чергу буде вимагати додаткового простору. Розділ розміром 10 Гбайт є розумним розміром для забезпечення росту системи. Новостворена ОС сама по собі не буде займати стільки простору. Він необхідний для забезпечення достатнього вільного місця для тимчасового зберігання. Компілювання пакетів може вимагати великих розмірів дискового простору, який буде відновленим після встановлення пакету.

Через те, що не завжди достатнього вільної Пам'яті Довільного Доступу (Random Access Memory — RAM) для процесу компіляції, буде хорошою ідеєю використати малий розділ диску як місце обміну між RAM і дисковим простором (так названа область підкачки — swap). Вона використовується ядром системи для того, щоб зберегти рідко використовувані дані і залишити більше пам'яті для активних процесів. Розділ swap для системи LFS можу бути таким самим який використовує встановлений сторонній дистрибутив, що в даному випадку відкидає необхідність в створенні нового.

Запустіть програму для розбивки диску, наприклад, такі як **cfdisk** чи **fdisk** з опцією командної стрічки, яка вказує ім'я жорсткого диску, на якому буде створено новий розділ — для прикладу /dev/hda для основного Integrated Drive Electronics (IDE) жорсткого диску. Створіть новий “рідний” розділ і область підкачки, якщо це необхідно. Будь-ласка зверніться до cfdisk(8) чи fdusk(8), якщо ви не знаєте як використовувати ці програми.

> Увага
>
> Для досвідчених користувачів можливі і інші схеми розділів. Нова система LFS може бути на програмному масиві RAID чи на логічному томі LVM. Однак, деякі з цих опцій вимагають програму initramfs, але це вже окрема велика тема. Ці методології розбиття не рекомендуються для початківців.

Запам'ятайте назву нового розділу (наприклад, hda5). Ця книга буде посилатись до нього як розділ LFS. Також необхідно запам'ятати назву розділу swap. Ці імена будуть потрібними для файлу /etc/fstab.

2.2.1. Інші питання про розділи

Прохання про пораду в розбитті розділів часто задаються на списках розсилки LFS. Це є дуже суб'єктивна тема. За замовчуванням для більшості дистрибутивів є використання усього вінчестера (жорсткого диску) за виключенням малої області підкачки (swap). Такий спосіб розбивки не є оптимальним для системи LFS

через деякі причини. Воно занижує гнучкість, робить обмін даних між багатьма дистрибутивами чи LFS системами більш складним, сповільнює процедуру резервного копіювання, і може псувати дисковий об'єм через неефективність структур файлових систем.

2.2.1.1. Кореневий розділ

Кореневий розділ системи LFS (не сплутувати з директорією /root) розміром 10 гігабайт є хорошим компромісом для більшості систем. Воно забезпечує достатню кількість простору для процесу побудови LFS системи і більшості систем BLFS, але достатньо малим для того щоб було легко створювати інші розділи для подальшого експериментування.

2.2.1.2. Область swap

Більшість дистрибутивів створюють область підкачки самостійно. Зазаичай рекомендований розмір області swap приблизно в двічі більший ніж фізичний розмір RAM, однак це зазвичай рідко потрібно. Якщо розмір жорсткого диску обмежений, тоді встановіть розмір області підкачки рівним двом гігабайтам і робіть моніторинг дискового свапінгу.

Свапінг є не завжди корисним. Зазвичай ви можете визначити, чи використовує система область підкачки, просто “підслуховуючи” дискову активність і переглядаючи як система реагує на команди. Першою реакцією на свапінг має бути перевірка на розумність команди такої як, наприклад, спроба редагувати файл розміром 5 гігабайт. Якщо свапінг стає нормальним явищем для системи, найкращим вирішенням цього буде придбати більшого об'єму RAM на вашу систему.

2.2.1.3. Інші зручні розділи

Я декілька інші розділи, які не вимагаються, але можуть бути враховані, коли створюється дискова розмітка. Наступний список не є повним, але його можна використати як гід.

- /boot — дуже рекомендований. Використовуйте цей розділ для зберігання ядер і іншої інформації завантаження. Для мінімізації потенційних проблем завантаженням з великими дисками, зробіть цей розділ фізично першим на вашому першому жорсткому диску. Розмір у 100 Мбайт буде цілком адекватний.

- /home — дуже рекомендовано. Поділяйте вашу домашню директорію і налаштування користувача між багатьма дистрибутивами чи побудованих LFS систем. Його розмір зазвичай дуже великий і залежить від доступного простору диску.

- /usr — Окремий розділ /usr зазвичай використовується у серверах для тонких клієнтів або станцій з малим розміром дискового простору. Він зазвичай не потрібний для LFS. Розмір у п'ять гігабайт буде нормальним для більшості систем.

- /opt — ця директорія використовується найбільше у BLFS, де багато встановлень великих пакетів, таких як Gnome чи KDE можуть бути встановленими без вбудування файлів у ієрархію /usr. Якщо він використовується, тоді розмір від 5 до 10 гігабайт зазвичай достатньо.

- /tmp Окрема директорія /tmp використовується рідко, але є корисною для конфігурації тонкого клієнту. Цей розділ, якщо використовується, зазвичай не потрібно перевищувати у декілька гігабайт.

- /usr/src — цей розділ є дуже корисним для забезпечення місця зберігання вихідного коду BLFS і розділення їх між побудовами LFS. Він може також використовуватися як місце для збирання пакетів

BLFS. Розумний розмір у 30-50 Гбайт забезпечує достатньо місця.

Будь-який окремий розділ, який ви хочете автоматично монтувати під час завантаження ОС, необхідно поміщати у файл /etc/fstab. Інформація про те як у ньому вказати розділи буде обговореною у Секції 8.2 “Створюючи файл /etc/fstab”.

2.3. Створення файлових систем на нових розділах

Якщо пустий розділ вже створено, можна форматувати його новою файловою системою. Найбільш широко використовуваною системою у світі Лінукс є друга розширена файлова система (ext2), але з новими високопродуктивними жорсткими дисками, журналюванні файлові системи стають все більш і більш популярними. Третя розширена файлова система (ext3) широко використовується як покращення до ext2, яка додає журнальні можливості і є сумісною з утилітою E2fsprogs. Ми створимо файлову систему ext3. Інструкції для створення інших файлових систем можна прочитати за адресою http://www.linuxfromscratch.org/blfs/view/svn/postlfs/filesystems.html.

Щоб відформатувати LFS розділ у файлову систему ext3, виконайте наступне:

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

Замініть <xxx> ім'ям розділу LFS (hda у нашому попередньому прикладі).

Увага

Деякі дистрибутиви використовують деякі свої особливості у їхньому інструментарії (E2fsprogs). Це може спричинити проблеми з завантаженням у нову систему в Частині 9, так як ці особливості не будуть підтримуватися встановленим у LFS програмою E2fsprogs; і ви отримаєте помилку подібну до “unsuported filesystem features, update your e2fsprogs”. Для перевірки того, чи ваша система використовує особливі поліпшення, виконайте наступну команду:

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

Якщо вивід вміщує параметри відмінні від has_journal, ext_attr, resize_inode, dir_index, filetype, sparse_super, large_file чи needs_recovery , тоді ваша система використовує певні відмінності. У цьому випадку, щоб запобігти майбутні проблеми, ви повинні скомпілювати інвентар пакету E2fsprogs і використовувати вихідні бінарні файли для перебудови файлових систем:

```
cd /tmp
tar -xzvf /path/to/sources/e2fsprogs-1.42.5.tar.gz
cd e2fsprogs-1.42.5
mkdir -v build
cd build
../configure
make #note that we intentionally don't 'make install' here!
./misc/mke2fs -jv /dev/<xxx>
cd /tmp
rm -rfv e2fsprogs-1.42.5
```

Якщо ви використовуєте існуючий розділ swap, немає необхідності його форматувати. Якщо був створений новий розділ для swap, необхідно ініціалізувати його цією командою:

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

Замініть <yyy> ім'ям розділу swap.

2.4. Підмотування нового розділу

Тепер, коли файлова система створена, необхідно зробити цей розділ доступним. Щоб це зробити, нам необхідно монтувати його під вибраною точкою монтування. У цілях даної книги, передбачено, що ця файлова система буде підключена у директорію /mnt/lfs, але вибір залишається між вами.

Виберіть точку монтування і прив'яжіть її до змінної середовища LFS, виконавши команду:

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

Опісля, створіть директорію монтування і підключіть розділ LFS командою:

```
mkdir -pv $LFS

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

Замініть <xxx> назвою розділу LFS.

Якщо ви використовуєте декілька розділів для системи LFS (тобто, один для / і інший для /usr), монтуйте їх виконуючи:

```
mkdir -pv $LFS

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

mkdir -v $LFS/usr

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

Замініть <xxx> і <yyy> відповідними іменами розділів.

Впевніться, що нові розділи не монтовані з обмежувальними правами (такі опції як nosuid або nodev). Виконайте команду mount без яких-небудь параметрів, щоб побачити які опції встановлені для підключених розділів LFS. Якщо nosuid, nodev і/або noatime увімкнені, розділи необхідно перемонтувати.

Якщо ви використовуєте розділ swap, впевніться, що він увімкнутий, використовуючи команду swapon:

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

Замініть <zzz> ім'ям розділу swap.

Тепер якщо є організоване місце для роботи, час завантажити пакети.

Глава 3. Пакети і патчі

3.1. Вступ

Дана частина вміщає список пакетів, які необхідно завантажити щоб побудувати базову систему Лінукс. Версії програм у списку відповідають версіям програмного забезпечення, відомого як те, що працює, і ця книга базована на їхньому використанні. Ми дуже рекомендуємо не використовувати нові версії тому, що команди до побудови можуть не працювати з ними. Новіші версії пакетів також можуть мати проблеми, які будуть вимагати певних обхідних шляхів. Ці шляхи будуть розроблені і стабілізовані у розроблюваній версії книги.

Адреси завантаження не завжди можуть бути доступними. Якщо сторінка завантаження змінила свою адресу після того як вийшла дана книга, Google (http://www.google.com) забезпечує хороший механізм пошуку для більшості пакетів. Якщо пошук не дав результатів, спробуйте один з альтернативних засобів завантаження, обговорених за адресою http://www.linuxfromscratch.org/lfs/packages.html#packages.

Завантажені пакети і патчі найкраще всього зберегти у місці, яке буде легко доступним для майбутньої побудови. Робоча директорія також буде необхідною для розпакування коду і збирання його. Використовуючи її, необхідні елементи будуть розміщені на розділі LFS і будуть доступними для усіх стадій процесу побудови.

Щоб створити даний каталог, виконайте наступну команду, від імені користувача root, перед тим як розпочати процес завантаження:

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

Зробіть дану директорію доступно для запису і ”липкою” (англ. Sticky). “Липка” означає, навіть якщо багато користувачів мають права запису на директорію, тільки власник файлу може видалити файл у липкій директорії. Наступна команда увімкне права запису і липкий біт:

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

Легким шляхом завантаження усіх пакетів і патчів є використання списку для команди wget. Приклад:

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

Додатково, розпочинаючи з LFS версії 7.0, є окремий файл, md5sums, який можна використати щоб перевірити чи усі коректні пакети доступні, перед ти як почати процес. Розмістіть цей файл у директорії $LFS/sources і виконайте:

```
pushd $LFS/sources

md5sum -c md5sums

popd
```

3.2. Пакети

Завантажте або отримайте іншим шляхом наступні пакети:

- **Autoconf (2.69) - 1,186 Кбайт:**

Домашня сторінка: *http://www.gnu.org/software/autoconf/*

Завантажити: *http://ftp.gnu.org/gnu/autoconf/autoconf-2.69.tar.xz*

Сума MD5: `50f97f4159805e374639a73e2636f22e`

• **Automake (1.12.3) - 1,352 КБайт:**

Домашня сторінка: *http://www.gnu.org/software/automake/*

Завантажити: *http://ftp.gnu.org/gnu/automake/automake-1.12.3.tar.xz*

Сума MD5: `0df082825f8f41087eb70c5088f4515e`

• **Bash (4.2) - 6,845 Кбайт:**

Домашня сторінка: *http://www.gnu.org/software/bash/*

Завантажити: *http://ftp.gnu.org/gnu/bash/bash-4.2.tar.gz*

Сума MD5: `3fb927c7c33022f1c327f14a81c0d4b0`

• **Binutils (2.22) - 19,505 Кбайт:**

Домашня сторінка: *http://www.gnu.org/software/binutils/*

Завантажити: *http://ftp.gnu.org/gnu/binutils/binutils-2.22.tar.bz2*

Сума MD5: ee0f10756c84979622b992a4a61ea3f5

• **Bison (2.6.2) - 1,612 Кбайт:**

Домашня сторінка: *http://www.gnu.org/software/bison/*

Завантажити: *http://ftp.gnu.org/gnu/bison/bison-2.6.2.tar.xz*

Сума MD5: `dea291996f98c34c3fd8e389a9cf6ea1`

• **Bzip2 (1.0.6) - 764 Кбайт:**

Домашня сторінка: *http://www.bzip.org/*

Завантажити: *http://www.bzip.org/1.0.6/bzip2-1.0.6.tar.gz*

Сума MD5: `00b516f4704d4a7cb50a1d97e6e8e15b`

• **Check (0.9.8) - 546 Кбайт:**

Домашня сторінка: *http://check.sourceforge.net/*

Завантажити: *http://sourceforge.net/projects/check/files/check/0.9.8/check-0.9.8.tar.gz*

Сума MD5: `5d75e9a6027cde79d2c339ef261e7470`

• **Coreutils (8.19) - 4,992 Кбайти:**

Домашня сторінка: *http://www.gnu.org/software/coreutils/*

Завантажити: *http://ftp.gnu.org/gnu/coreutils/coreutils-8.19.tar.xz*

Сума MD5: `1a01231a2f3ed37c0efc073ccdda9375`

• **DejaGNU (1.5) - 563 Кбайт:**

Домашня сторінка: *http://www.gnu.org/software/dejagnu/*

Завантажити: *http://ftp.gnu.org/gnu/dejagnu/dejagnu-1.5.tar.gz*

Сума MD5: `3df1cbca885e751e22d3ebd1ac64dc3c`

• **Diffutils (3.2) - 1,976 Кбайт:**

Домашня сторінка: *http://www.gnu.org/software/diffutils/*

Завантажити: *http://ftp.gnu.org/gnu/diffutils/diffutils-3.2.tar.gz*

Сума MD5: `22e4deef5d8949a727b159d6bc65c1cc`

• **E2fsprogs (1.42.5) - 5,780 Кбайт:**

Домашня сторінка: *http://e2fsprogs.sourceforge.net/*

Завантажити: *http://prdownloads.sourceforge.net/e2fsprogs/e2fsprogs-.42.5.tar.gz*

Сума MD5: `aca828bb4bcca20991a442deb950b670`

• **Expect (5.45) - 614 Кбайт:**

Домашня сторінка: *http://expect.sourceforge.net/*

Завантажити: *http://prdownloads.sourceforge.net/expect/expect5.45.tar.gz*

Сума MD5: `44e1a4f4c877e9ddc5a542dfa7ecc92b`

• **File (5.11) - 596 Кбайт:**

Домашня сторінка: *http://www.darwinsys.com/file/*

Завантажити: *ftp://ftp.astron.com/pub/file/file-5.11.tar.gz*

Сума MD5: `16a407bd66d6c7a832f3a5c0d609c27b`

> Увага
>
> Fike (5.11) може бульше не бути доступним під адресою у списку. Адміністрація сайту, видаляють старі версії пакетів, коли виходить нова. Альтернативна адреса завантаження, яка буде мати необхідну версію є доступна за адресою http://www. linuxfromscratch.org/lfs/download.html#ftp.

• **Findutils (4.4.2) - 2,100 Кбайт:**

Домашня сторінка: *http://www.gnu.org/software/findutils/*

Завантажено: *http://ftp.gnu.org/gnu/findutils/findutils-4.4.2.tar.gz*

Сума MD5: `351cc4adb07d54877fa15f75fb77d39f`

• **Flex (2.5.37) - 1,280 Кбайт:**

Домашня сторінка: *http://flex.sourceforge.net*

Завантажити: *http://prdownloads.sourceforge.net/flex/flex-2.5.37.tar.bz2*

Сума MD5: `c75940e1fc25108f2a7b3ef42abdae06`

• **Gawk (4.0.1) - 1,575 Кбайт:**

Домашня сторінка: *http://www.gnu.org/software/gawk/*

Завантажити: *http://ftp.gnu.org/gnu/gawk/gawk-4.0.1.tar.xz*

Сума MD5: `a601b032c39cd982f34272664f8afa49`

• **GCC (4.7.1) - 80,703 Кбайт:**

Домашня сторінка: *http://gcc.gnu.org/*

Завантажити: *http://ftp.gnu.org/gnu/gcc/gcc-4.7.1/gcc-4.7.1.tar.bz2*

Сума MD5: `933e6f15f51c031060af64a9e14149ff`

• **GDBM (1.10) - 640 Кбайт:**

Домашня сторінка: *http://www.gnu.org/software/gdbm/*

Завантажити: *http://ftp.gnu.org/gnu/gdbm/gdbm-1.10.tar.gz*

Сума MD5: `88770493c2559dc80b561293e39d3570`

• **Gettext (0.18.1.1) - 14,785 Кбайт:**

Домашня сторінка: *http://www.gnu.org/software/gettext/*

Завантажити: *http://ftp.gnu.org/gnu/gettext/gettext-0.18.1.1.tar.gz*

Сума MD5: `3dd55b952826d2b32f51308f2f91aa89`

• **Glibc (2.16.0) - 9,756 Кбайт:**

Домашня сторінка: *http://www.gnu.org/software/libc/*

Завантажити: *http://ftp.gnu.org/gnu/glibc/glibc-2.16.0.tar.xz*

Сума MD5: `80b181b02ab249524ec92822c0174cf7`

• **GMP (5.0.5) - 1,632 Кбайт:**

Домашня сторінка: *http://www.gnu.org/software/gmp/*

Завантажити: *http://ftp.gnu.org/gnu/gmp/gmp-5.0.5.tar.xz*

Сума MD5: `8aef50959acec2a1ad41d144ffe0f3b5`

• **Grep (2.14) - 1,172 Кбайт:**

Домашня сторінка: *http://www.gnu.org/software/grep/*

Завантажити: *http://ftp.gnu.org/gnu/grep/grep-2.14.tar.xz*

Сума MD5: `d4a3f03849d1e17ce56ab76aa5a24cab`

• Groff (1.21) - 3,774 Кбайт:

Домашня сторінка: *http://www.gnu.org/software/groff/*

Завантажити: *http://ftp.gnu.org/gnu/groff/groff-1.21.tar.gz*

Сума MD5: `8b8cd29385b97616a0f0d96d0951c5bf`

• GRUB (2.00) - 5,016 Кбайт:

Домашня сторінка: *http://www.gnu.org/software/grub/*

Завантажити: *http://ftp.gnu.org/gnu/grub/grub-2.00.tar.xz*

Сума MD5: `a1043102fbc7bcedbf53e7ee3d17ab91`

• Gzip (1.5) - 704 Кбайт:

Домашня сторінка: *http://www.gnu.org/software/gzip/*

Завантажити: *http://ftp.gnu.org/gnu/gzip/gzip-1.5.tar.xz*

Сума MD5: `2a431e169b6f62f7332ef6d47cc53bae`

• Iana-Etc (2.30) - 201 Кбайт:

Домашня сторінка: *http://freshmeat.net/projects/iana-etc/*

Завантажити: *http://anduin.linuxfromscratch.org/sources/LFS/lfs-packages/conglomeration//iana-etc/iana-etc-2.30.tar.bz2*

Сума MD5: `3ba3afb1d1b261383d247f46cb135ee8`

• Inetutils (1.9.1) - 1,941 KB:

Домашня сторінка: *http://www.gnu.org/software/inetutils/*

Заватажити: *http://ftp.gnu.org/gnu/inetutils/inetutils-1.9.1.tar.gz*

Сума MD5: `944f7196a2b3dba2d400e9088576000c`

• IPRoute2 (3.5.1) - 379 Кбайт:

Домашня сторінка: *http://www.kernel.org/pub/linux/utils/net/iproute2/*

Завантажити: *http://www.kernel.org/pub/linux/utils/net/iproute2/iproute2-3.5.1.tar.xz*

Сума MD5: `d4425b44edd5eacd6099e672e4baacbf`

• Kbd (1.15.3) - 1,621 Кбайт:

Домашня сторінка: *http://ftp.altlinux.org/pub/people/legion/kbd*

Завантажити: *http://ftp.altlinux.org/pub/people/legion/kbd/kbd-1.15.3.tar.gz*

Сума MD5: `8143e179a0f3c25646ce5085e8777200`

• Kmod (9) - 1,096 Кбайт:

Завантажити: *http://www.kernel.org/pub/linux/utils/kernel/kmod/kmod-9.tar.xz*

Сума MD5: `c8ae2d2694fbca2b28e238b30543a0cd`

• Less (444) - 301 Кбайт:

Домашня сторінка: *http://www.greenwoodsoftware.com/less/*

Завантажити: *http://www.greenwoodsoftware.com/less/less-444.tar.gz*

Сума MD5: `56f9f76ffe13f70155f47f6b3c87d421`

• LFS-Bootscripts (20120901) - 33 Кбайт:

Завантажити: *http://www.linuxfromscratch.org/lfs/downloads/7.2/lfs-bootscripts-20120901.tar.bz2*

Сума MD5: `393e4ed76819ce412f8a406c44beabd0`

• Libpipeline (1.2.1) - 723 Кбайт:

Домашня сторінка: *http://libpipeline.nongnu.org/*

Завантажити: *http://download.savannah.gnu.org/releases/libpipeline/libpipeline-1.2.1.tar.gz*

Сума MD5: `20896c919eca4acb3d2f13012fb7ba02`

• Libtool (2.4.2) - 2,571 Кбайт:

Домашня сторінка: *http://www.gnu.org/software/libtool/*

Завантажити: *http://ftp.gnu.org/gnu/libtool/libtool-2.4.2.tar.gz*

Сума MD5: `d2f3b7d4627e69e13514a40e72a24d50`

• Linux (3.5.2) - 66,060 Кбайт:

Домашня сторінка: *http://www.kernel.org/*

Завантажити: *http://www.kernel.org/pub/linux/kernel/v3.x/linux-3.5.2.tar.xz*

Сума MD5: `b3cfccfb6961ea407acf0b070184b0b1`

> Увага
>
> Ядро Лінукс поновлюється відносно часто, багато разів через те, що виявлено проблеми безпеки.
>
> Остання доступна версія ядра 3.5.x може бути використаною, якщо не заперечує сторінка помилок.
>
> Для користувачів, з обмеженою швидкістю чи дорогим трафіком, які бажають оновити ядро, основна лінія версій пакетів і патчів можуть бути завантажені окремо. Це може зберегти деякий час і гроші для наступного оновлення версії патчів у мінорних випусках.

• **M4 (1.4.16) - 1,229 Кбайт:**

Домашня сторінка: *http://www.gnu.org/software/m4/*

Завантажити: *http://ftp.gnu.org/gnu/m4/m4-1.4.16.tar.bz2*

Сума MD5: `8a7cef47fecab6272eb86a6be6363b2f`

• **Make (3.82) - 1,213 Кбайт:**

Домашня сторінка: *http://www.gnu.org/software/make/*

Завантажити: *http://ftp.gnu.org/gnu/make/make-3.82.tar.bz2*

Сума MD5: `1a11100f3c63fcf5753818e59d63088f`

• **Man-DB (2.6.2) - 1,353 Кбайт:**

Домашня сторінка: *http://www.nongnu.org/man-db/*

Завантажити: *http://download.savannah.gnu.org/releases/man-db/man-db-2.6.2.tar.xz*

Сума MD5: `647c48d46c464419185d031d04481ee5`

• **Man-pages (3.42) - 1,076 Кбайт:**

Домашня сторінка: *http://www.kernel.org/doc/man-pages/*

Завантажити: *http://www.kernel.org/pub/linux/docs/man-pages/man-pages-3.42.tar.xz*

Сума MD5: `2392bb23db94f344f493c4beca41398f`

• **MPC (1.0) - 614 Кбайт:**

Домашня сторінка: *http://www.multiprecision.org/*

Завантажити: *http://www.multiprecision.org/mpc/download/mpc-1.0.tar.gz*

Сума MD5: `13370ceb2e266c5eeb2f7e78c24b7858`

• **MPFR (3.1.1) - 1,047 Кбайт:**

Домашня сторінка: *http://www.mpfr.org/*

Завантажити: *http://www.mpfr.org/mpfr-3.1.1/mpfr-3.1.1.tar.xz*

Сума MD5 `91d51c41fcf2799e4ee7a7126fc95c17`

• **Ncurses (5.9) - 2,760 Кбайт:**

Домашня сторінка: *http://www.gnu.org/software/ncurses/*

Завантажити: *ftp://ftp.gnu.org/gnu/ncurses/ncurses-5.9.tar.gz*

Сума MD5: `8cb9c412e5f2d96bc6f459aa8c6282a1`

• **Patch (2.6.1) - 248 Кбайт:**

Домашня сторінка: *http://savannah.gnu.org/projects/patch/*

Завантажити: *http://ftp.gnu.org/gnu/patch/patch-2.6.1.tar.bz2*

Сума MD5: `0818d1763ae0c4281bcdc63cdac0b2c0`

• **Perl (5.16.1) - 13,256 Кбайт:**

Домашня сторінка: *http://www.perl.org/*

Завантажити: *http://www.cpan.org/src/5.0/perl-5.16.1.tar.bz2*

Сума MD5: `b87358e2c461a898cfd7c334e7dd8993`

• **Pkg-config (0.27) - 1872 Кбайт:**

Домашня сторінка: *http://www.freedesktop.org/wiki/Software/pkg-config*

Завантажити: *http://pkgconfig.freedesktop.org/releases/pkg-config-0.27.tar.gz*

Сума MD5: `3a4c9feab14b6719afd8904945d9b4e4`

• **Procps (3.2.8) - 279 Кбайт:**

Домашня сторінка: *http://procps.sourceforge.net/*

Завантажити: *http://procps.sourceforge.net/procps-3.2.8.tar.gz*

Сума MD5: `9532714b6846013ca9898984ba4cd7e0`

• **Psmisc (22.19) - 481 Кбайт:**

Домашня сторінка: *http://psmisc.sourceforge.net/*

Завантажити: *http://prdownloads.sourceforge.net/psmisc/psmisc-22.19.tar.gz*

Сума MD5: `38563b4760ffce54b0eadf99cb5b16e8`

• **Readline (6.2) - 2,225 Кбайт:**

Домашня сторінка: *http://cnswww.cns.cwru.edu/php/chet/readline/rltop.html*

Завантажити: *http://ftp.gnu.org/gnu/readline/readline-6.2.tar.gz*

Сума MD5: `67948acb2ca081f23359d0256e9a271c`

• **Sed (4.2.1) - 878 Кбайт:**

Домашня сторніка: *http://www.gnu.org/software/sed/*

Завантажити: *http://ftp.gnu.org/gnu/sed/sed-4.2.1.tar.bz2*

Сума MD5: `7d310fbd76e01a01115075c1fd3f455a`

• **Shadow (4.1.5.1) - 2,142 Кбайт:**

Домашня сторінка: *http://pkg-shadow.alioth.debian.org/*

Завантажено: *http://pkg-shadow.alioth.debian.org/releases/shadow-4.1.5.1.tar.bz2*

Сума MD5: `a00449aa439c69287b6d472191dc2247`

• Sysklogd (1.5) - 85 Кбайт:

Домашня сторінка: *http://www.infodrom.org/projects/sysklogd/*

Завантажити: *http://www.infodrom.org/projects/sysklogd/download/sysklogd-1.5.tar.gz*

Сума MD5: `e053094e8103165f98ddafe828f6ae4b`

• Sysvinit (2.88dsf) - 108 Кбайт:

Домашня сторінка: *http://savannah.nongnu.org/projects/sysvinit*

Завантажити: *http://download.savannah.gnu.org/releases/sysvinit/sysvinit-2.88dsf.tar.bz2*

Сума MD5: `6eda8a97b86e0a6f59dabbf25202aa6f`

• Tar (1.26) - 2,285 Кбайт:

Домашня сторінка: *http://www.gnu.org/software/tar/*

Завантажити: *http://ftp.gnu.org/gnu/tar/tar-1.26.tar.bz2*

Сума MD5: `2cee42a2ff4f1cd4f9298eeeb2264519`

• Tcl (8.5.12) - 4,396 Кбайт:

Домашня сторінка: *http://tcl.sourceforge.net/*

Завантажити: *http://prdownloads.sourceforge.net/tcl/tcl8.5.12-src.tar.gz*

Сума MD5: `174b2b4c619ba8f96875d8a051917703`

• Time Zone Data (2012e) - 208 Кбайт:

Домашня сторінка: *http://www.iana.org/time-zones*

Завантажити: *http://www.iana.org//time-zones/repository/releases/tzdata2012e.tar.gz*

Сума MD5: `cb74e1f7bcc9a968a891a471e72e47b8`

• Texinfo (4.13a) - 2,687 Кбайт:

Домашня сторінка: *http://www.gnu.org/software/texinfo/*

Завантажити: *http://ftp.gnu.org/gnu/texinfo/texinfo-4.13a.tar.gz*

Сума MD5: `71ba711519209b5fb583fed2b3d86fcb`

• Systemd (188) - 1,324 Кбайт:

Домашня сторінка: *http://www.freedesktop.org/wiki/Software/systemd/*

Завантажити: *http://www.freedesktop.org/software/systemd/systemd-188.tar.xz*

Сума MD5: d89b42699695554949d072ef46c0dfc9

• **Udev-lfs Tarball (188) - 20 Кбайт:**

Завантажити: *http://anduin.linuxfromscratch.org/sources/other/udev-lfs-188-3.tar.bz2*

Сума MD5: ef6cd9f078c39c61ba744d08276a1210

• **Util-linux (2.21.2) - 2,916 КБайт:**

Домашня сторінка: *http://userweb.kernel.org/~kzak/util-linux/*

Завантажити: *http://www.kernel.org/pub/linux/utils/util-linux/v2.21/util-linux-2.21.2.tar.xz*

Сума MD5: 54ba880f1d66782c2287ee2c898520e9

• **Vim (7.3) - 8,675 Кбайт:**

Домашня сторінка: *http://www.vim.org*

Завантажити: *ftp://ftp.vim.org/pub/vim/unix/vim-7.3.tar.bz2*

Сума MD5: 5b9510a17074e2b37d8bb38ae09edbf2

• **Xz Utils (5.0.4) - 894 Кбайт:**

Домашня сторінка: *http://tukaani.org/xz*

Завантажити: *http://tukaani.org/xz/xz-5.0.4.tar.xz*

Сума MD5: 161015c4a65b1f293d31810e1df93090

• **Zlib (1.2.7) - 493 Кбайт:**

Домашня сторінка: *http://www.zlib.net/*

Завантажити: *http://www.zlib.net/zlib-1.2.7.tar.bz2*

Сума MD5: 2ab442d169156f34c379c968f3f482dd

Сумарний розмір пакетів складає приблизно 292 Мбайт.

3.3. Необхідні патчі

В добавку до пакетів, необхідними також є декілька патчів. Ці патчі виправляють деякі помилки в пакетах, які повинні бути виправленими підтримкою. Також, ці патчі роблять невеликі модифікації в пакетах, для того щоб з ними було легше працювати. Наступні патчі будуть необхідними, щоб зібрати систему LFS.

• **Bash Upstream Fixes Patch - 51 Кбайт:**

Завантажити: *http://www.linuxfromscratch.org/patches/lfs/7.2/bash-4.2-fixes-8.patch*

Сума MD5: e82d2200e82aa28640299bbcad140361

• **Binutils Build Fix Patch - 1.4 Кбайт:**

Завантажити: *http://www.linuxfromscratch.org/patches/lfs/7.2/binutils-2.22-build_fix-1.patch*

Сума MD5: ddc5a9a170ed6ba23b8eb7d808e609ee

• **Bzip2 Documentation Patch - 1.6 Кбайт:**

Завантажити: *http://www.linuxfromscratch.org/patches/lfs/7.2/bzip2-1.0.6-install_docs-1.patch*

Сума MD5: 6a5ac7e89b791aae556de0f745916f7f

• **Coreutils Internationalization Fixes Patch - 272 Кбайт:**

Завантажити: *http://www.linuxfromscratch.org/patches/lfs/7.2/coreutils-8.19-i18n-1.patch*

Сума MD5: befbf82628d019ef08d522c95cba331d

• **Flex Regression Tests Patch - 2.8 Кбайт:**

Завантажити: *http://www.linuxfromscratch.org/patches/lfs/7.2/flex-2.5.37-bison-2.6.1-1.patch*

Сума MD5: d5b001ef9bdbbe32e2f27576d97d8ff0

• **Glibc DNS Resolve Patch - 2.0 Кбайт:**

Завантажити: *http://www.linuxfromscratch.org/patches/lfs/7.2/glibc-2.16.0-res_query_fix-1.patch*

Сума MD5: d37659c643b1a2150624120238e5b295

• **Kbd Loadkeys Fix Patch - 1.6 Кбайт:**

Завантажити: *http://www.linuxfromscratch.org/patches/lfs/7.2/kbd-1.15.3-upstream_fixes-1.patch*

Сума MD5: 58ae9bd7d546426cfaccf3eba16ad1a2

• **Kbd Backspace/Delete Fix Patch - 12 Кбайт:**

Завантажити: *http://www.linuxfromscratch.org/patches/lfs/7.2/kbd-1.15.3-backspace-1.patch*

Сума MD5: f75cca16a38da6caa7d52151f7136895

• **Kmod Testsuite Patch - 2.2 Кбайт:**

Завантажити: *http://www.linuxfromscratch.org/patches/lfs/7.2/kmod-9-testsuite-1.patch*

Сума MD5: 11ab14f5b63ae3c163804517cf110fbb

• **Make Upstream Fixes Patch - 9.7 Кбайт:**

Завантажити: *http://www.linuxfromscratch.org/patches/lfs/7.2/make-3.82-upstream_fixes-2.patch*

Сума MD5: 02c0f3989185a7345233872b1ae5f46d

• **Patch Testsuite Fix Patch - 1 Кбайт:**

Завантажити: *http://www.linuxfromscratch.org/patches/lfs/7.2/patch-2.6.1-test_fix-1.patch*

Сума MD5: c51e1a95bfc5310635d05081472c3534

• **Perl Libc Patch - 1.6 Кбайт:**

Завантажити: *http://www.linuxfromscratch.org/patches/lfs/7.2/perl-5.16.1-libc-2.patch*

Сума MD5: `daf5c64fd7311e924966842680535f8f`

• **Procps HZ Errors Patch - 2.3 Кбайт:**

Завантажити: *http://www.linuxfromscratch.org/patches/lfs/7.2/procps-3.2.8-fix_HZ_errors-1.patch*

Сума MD5: `2ea4c8e9a2c2a5a291ec63c92d7c6e3b`

• **Procps Watch Patch - 3.5 KB:**

Завантажити: *http://www.linuxfromscratch.org/patches/lfs/7.2/procps-3.2.8-watch_unicode-1.patch*

Сума MD5: `cd1a757e532d93662a7ed71da80e6b58`

• **Readline Upstream Fixes Patch - 1.3 Кбайт:**

Завантажити: *http://www.linuxfromscratch.org/patches/lfs/7.2/readline-6.2-fixes-1.patch*

Сума MD5: `3c185f7b76001d3d0af614f6f2cd5dfa`

• **Sed Regression Tests Patch - 1.9 Кбайт:**

Завантажити: *http://www.linuxfromscratch.org/patches/lfs/7.2/sed-4.2.1-testsuite_fixes-1.patch*

Сума MD5: `2c10a5804eedf5359bcf427bc0d05579`

Загальна сума патчів складає приблизно 367.9 Кбайт

В додаток до необхідних патчів, існує певна кількість необов'язкових, які створені громадою LFS. Ці необов'язкові патчі вирішують незначні проблеми або вмикають функціональність, яка не увімкнена за замовчуванням. Ви можете вільно прочитати базу даних цих патчів, яка розміщена за адресою http://www.linuxfromscratch.org/patches/downloads/ і отримуйте будь-який патч який допоможе вдовольнити ваші потреби.

Глава 4. Фінальна підготовка

4.1. Змінна середовища $LFS

Змінна середовища `$LFS` широко використовується у даній книзі. Це є обов'язковим, щоб ця змінна була завжди визначена. Вона повинна бути ініціалізованою значенням шляху до вибраної точки монтування розділу, де буде міститись система LFS (у нашому випадку `$LFS` буде рівна `/mnt/lfs`, якщо ви вибрали шлях, який вказаний у книзі). Перевірте правильність змінної `LFS`, виконавши команду:

```
echo $LFS
```

Перевірте правильність виводу результату виконання команди. Якщо вивід неправильний (не відповідає шляху до вами вибраної точки монтування), змінна може бути встановленою командою:

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

Маючи цю змінну встановленою, ви робите набір у таких командах як **mkdir $LFS/tools** оптимізованим. Командний інтерпретатор зробить автоматичну заміну послідовності літер “$LFS” на встановлене їй відповідне значення “/mnt/lfs” (чи на інший встановлений вами шлях), коли команда буде оброблятися.

Не забувайте перевіряти що змінна $LFS встановлена, коли ви залишаєте і входите заново до поточного робочого середовища. (наприклад, коли ви виконуєте команду **su** для входу в систему, як інший користувач).

4.2. Створюємо директорію $LFS/tools

Усі програми у Глава 5 будуть встановлені у директорію `$LFS/tools` для того, щоб зберегти окремо програми, які будуть скомпільовані у Частині 6. Програми скомпільовані у цій директорії є тимчасовим інструментарієм і не він не буде частиною кінцевої системи LFS. Тримаючи ці програми в окремій директорії, ви можете легко видалити їх після використання. Це також запобігає змішування їх з програмами вихідної системи.

Створіть необхідну директорію, виконуючи наступну команду як користувач root:

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

Наступним кроком буде створення посилання /tools у кореневій директорії встановленого стороннього дистрибутиву. Це буде вказівником до новоствореної директорії у розділі для системи LFS. Виконайте дану нижче команду:

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

> Увага
>
> Дана вище команда є коректною. Команда **ln** має декілька синтаксичних варіацій, отож, будьте перевірте **info coreutils ln** і ln(1) перед тим, як повідомляти про те що ви вважаєте помилкою.

4.3. Додаємо користувача LFS

Коли ви знаходитесь в системі, як користувач root, одна помилка може спричинити пошкодження чи руйнування системи. Отож, ми рекомендуємо збирати усі пакети у цій частині, як непривілейований користувач Ви можете використати ваше власне ім'я користувача, але щоб зробити легшим встановлення чистого робочого середовища, створіть нового користувача, який називається lfs, як члена нової групи (також названої lfs) і використовуйте даний профіль для процесу встановлення. Як користувач root, виконайте дані команди для додавання нового користувача в систему:

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

Значення ключів (параметрів) команди:

-s /bin/bash

Цей ключ (параметр) вказує використовувати **bash** як основний командний інтерпретатор для користувача lfs.

-g lfs

Цей параметер додає користувача lfs до групи lfs.

-m

Це сворює домашню директорію для користувача lfs.

-k /dev/null

Цей параметер запобігає можливе копіювання файлів з скелетної директорії (зазвичай це `/etc/skel`) шляхом зміни шляху вводу до спеціального нульового пристрою.

lfs

Це є фактичне ім'я для новоствореного користувача і групи.

Для того щоб ввійти в систему як користувач lfs (в протилежність тому, щоб входити в систему як користувач lfs з профілю суперкористувача, що не вимагає введення паролю), дайте пароль користувачу lfs:

```
passwd lfs
```

Надайте користувачу lfs повний доступ до директорії $LFS/tools, роблячи його власником.

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

Якщо була створена окрема директорія, як і пропонувалося, дайте її у власність користувача lfs.

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

Наступним ділом, буде вхід в систему як користувач lfs. Це може бути виконане за допомогою віртуальної консолі, через віконний менеджер, або виконанням наступної команди яка заміняє користувача:

```
su - lfs
```

Ключ "-" вказує команді **su** розпочати сеанс командного інтерпретатора з реєстрацією замість того щоб розпочати сеанс без реєстрації. Різниця між цими двома типами сеансів може бути знайденою в деталях документації bash(1) і info bash.

4.4. Встановлення змінних середовища.

Встановимо хороше значення змінних робочого середовища, створюючи два нових файли конфігурації для командного інтерпретатора bash. Будучи в системі як користувач `lfs`, виконайте наступні команди для створення нового файлу `.bash_profile`:

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

Коли ви входите в систему як користувач `lfs`, поточна оболонка є і вхідною, це означає, що вона читає файл конфігурації `/etc/profile` дистрибутиву (який зазвичай містить деякі налаштування змінних

середивища) і після цього `.bash_profile`. Команда **exec env -i ... /bin/bahs** у файлі `.bash_profile` заміняє поточний командний інтепретатор новим, з повністю пустими змінними середовища, звичайно крім змінних HOME, TERM і PS1. Це запобігає вплив непотрібних і потенційно небезпечних змінних середовища на процес побудови системи. Техніка яка тут використовується, досягає цілі забезпечення чистого операційного середовища.
Новою інстанцією командного інтерпретатора є інтерпретатор без ідентифікації, який не читає файли `/etc/profile` і `.bash_profile`, а скоріше читає `.bashrc`. Тепер створіть файл `.bashrc` з змістом:

```
cat > ~/.bashrc << "EOF"
set +h
umask 022
LFS=/mnt/lfs
LC_ALL=POSIX
LFS_TGT=$(uname -m)-lfs-linux-gnu
PATH=/tools/bin:/bin:/usr/bin
export LFS LC_ALL LFS_TGT PATH
EOF
```

Команда **set +h** вимикає функцію хешування у **bash.** Хеш зазвичай є хорошою можливістю — **bash** використовує хеш-таблиці щоб запам'ятати повний шлях до виконуваних файлів, для того щоб запобігти пошук їх у шляхах, вказаних змінною `PATH`, і знаходженню цих самих програм. Однак, нові інструменти повинні використовуватись вже після того, як вони встановлені. Вимикаючи хеш-функцію, командний інтерпретатор буде завжди шукати шлях до програми, як тільки ви хочете виконати її. В результаті, командний інтерпретатор знайде новий скомпільований інструментарій у директорії `$LFS/tools` тоді як вони стануть доступними, без запам'ятовування старіших версій цих самих програм, розміщених у другому місці.

Встановлюючи маску створення файлів користувача (umask) у значення 022 запевняє, що новостворені файли і директорії можуть бути перезаписані їхнім власником, але інші користувачі можуть читати і запускати на виконання програми (припускаючи, що звичайні права використовуються системним викликом `open(2)`, нові файли будуть створенні з правами доступу 644, а директорії з правами 755).

Змінна `LFS` повинна мати в собі рядок шляху до вашої вибраної точки монтування LFS.

Змінна середовища `LC_ALL` контролює локалізацію певних програм, і робить їхні повідомлення слідувати правилам вказаної країни. Якщо хостова система використовує версію Glibc старішу ніж 2.2.4, інше значення цієї змінної ніж “POSIX” чи “C” може привести до деяких проблем після нового входу в систему. Встановлюючи змінну `LC_ALL` в значення “POSIX” чи “C” (значення еквівалентні) забезпечить роботу системи так як це очікується у середовищі після виконанні команди chroot.

Змінна `LFS_TGT` встановлює нестандартну, але сумісний машинний опис для використання поки компілюється наш крос компілятор і компонувальник. Більше інформації можна найти в Секції 5.2 “Технічні нотатки інструментарію”.

Встановлюючи `/tools/bin/` в переді стандартної змінної `PATH`, усі програми встановлені в Частині 5 виконуються командним інтерпретатором негайно після їхнього встановлення. Це, в поєднанні з хешуванням, обмежує ризик використання старих програм хостової системи, коли ті самі програми робочого середовища є доступними у Секції 5.

На кінець, маючи повністю підготовлене робоче середовище для побудови тимчасових інструментів, задайте щойно створену конфігурацію профілю:

```
source ~/.bash_profile
```

4.5. SBU

Багато людей перш за все хотіли б знати приблизно скільки займе часу компіляція і встановлення усіх пакетів. Через те, що Лінукс з Початків може бути побудованим на різних машинах, це є неможливим визначити точну кількість необхідного часу. Найбільший пакет (Glibc) займе 20 хвилин на швидких машинах, але може зайняти три дні на повільних! Замість того, щоб вказувати сам час, буде впроваджена нова одиниця часу — Standart Build Unit (SBU — Стандартна Одиниця Побудови).

Ця одиниця вимірюється наступним чином. Перший пакет, який буде скомпільований у цій книзі є Binutils у Частині 5. Час, який витрачається на його компілювання, це те що буде називатись SBU. Увесь інший час буде виражений відносно нього.

Для прикладу, візьмемо до уваги пакет, час компіляції якого буде складати час у 4.5 SBU. Це означає, що якщо система зайняла 10 хвилин часу на те щоб скомпілювати і встановити пакет Binutils, це займе приблизно 45 хвилин, щоб побудувати взятий пакет. На щастя, більшість побудов займе менше часу ніж побудова Binutils.

В загальному, SBU не є повністю точною одиницею тому, що вона залежить від багатьох факторів, включаючи версію GCC. Вона введена тут, щоб дати оцінку того, як довго може тривати встановлення пакету, але числа можуть сильно варіюватися у деяких випадках.

Для того, щоб побачити фактичні витрати часу для деякого числа машин, ми рекомендуємо переглянути домашню сторінку LinuxFromScratch SBU, яка знаходиться за адресою *http://www.linuxfromscratch.org/~sbu/*.

Увага

Для багатьох сучасних систем з декількома процесорами (чи ядрами) час компілювання для пакету можу бути занижений виконуючи паралельне сомпілювання (“parallel make”) встановлюючи змінну середовища або вказуючи команді **make** скільки процесорів є доступною у вас. Для прикладу, процесор Core2Duo може підтримувати два одночасних процеси побудови за допомогою:

```
Export MAKEFLAGS='-j 2'
```

або будуючи з параметрами:

```
make -j2
```

Коли таким шляхом задіяно багато процесорів, одиниця SBU буде часто варіюватися навіть більше ніж зазвичай. Аналізуючи вивід процесу побудови буде також складним через те, що рядки різних процесів будуть чергуватися. Якщо ви зустрілися з проблемою круку побудови, поверніться до однопроцесорної побудови для того, щоб правильно аналізувати повідомлення про помилку.

4.6. Тестові набори

Більшість пакетів постачають набір програм тестування. Виконуючи програму тестування для новозбудованої програми є хорошою ідеєю через те, що вони можуть забезпечувати “розумну перевірку”, вказуючи на те, що все скомпілювалося правильно. Набір програм для тестування, який проходить встановлені перевірки зазвичай доказує, що пакет працює так, як задумав розробник. Але це не гарантує, однак, що цей пакет є повністю вільним від семантичних помилок.

Деякі набори для тестування є важливіші ніж інші. Для прикладу, набір для тестування ядра пакета інструментарію GCC, Binutils і Glibc — є крайнє важливим через їхню центральну роль у правильному

функціонуванні системи. Тестові набори для GCC і Glibc можуть забрати багато часу для завершення, особливо на повільному апаратному забезпеченні, але є високо рекомендованими.

> Увага
>
> Досвідом показано, що є причини відмовитись від виконання тестів у Частині 5. Цьому служить факт того, що ви не можете втекти від впливу хостової системи на тести в цій частині, що часто спричиняє неочікувані помилки. Тому, що інструментарій, побудований в Частині 5 є тимчасовим і в кінцевому рахунку відкидається, ми не рекомендуємо вам виконувати тести в Частині 5. Інструкції для виконання цих тестів подаються для тестерів і розробників, але вони є необов'язковими.

Поширеною проблемою з виконуванням набору тестів для Binutils і GCC є виконування через псевдо термінали (PTYs). Це може спричинити велику кількість провалених тестів. Це може статись через декілька причин, але найбільш ймовірно, що це спричинено тим, що хостова система немає правильно встановленої файлової системи devpts. Це питання обговорюється у більших деталях за адресою http://www.linuxfromscratch.org//lfs/faq.html#no-ptys.

Інколи тестові набори пакетів провалюються, але це через причини які розробникам відомі і вважаються не критичними. Проконсультуйтеся журналом який розміщений за адресою http://www.linuxfromscratch.org/lfs/build-logs/7.2/ щоб перевірити чи ці помилки є очікуваними. Цей сайт є дійсним для усіх тестів в даній книзі.

Глава 5 Будівництво тимчасової системи.

5.1. Вступ

Ця частина показує як побудувати мінімальну систему Лінукс. Ця система буде вміщати тільки достатні інструменти для початку будування фінальної системи LFS у Главі 6, яка в свою чергу забезпечує більш зручну систему ніж мінімальний інструментарій.

Є два кроки у побудові мінімальної системи. Перший крок полягає в тому, щоб побудувати новий і незалежний від стороннього дистрибутиву інструментарій для побудови інших необхідних інструментів.

Файли, скомпільовані в цій частині, будуть встановленими у директорії `$LFS/tools`, для того щоб відділити їх від файлів які будуть встановленими у наступних частинах цієї книги. Через те, що пакети скомпільовані тут є тимчасовими, ми не хочемо, щоб вони засмічували майбутню систему LFS.

5.2. Технічні нотатки інструментарію

Ця секція роз'яснює деякі міркування і технічних деталі за загальним методом побудови. Це не є необхідним у негайному розумінні всього в цій секції. Більшість цієї інформації буде зрозумілішою після самого процесу побудови. Ви можете звертатися до цієї секції у любий час під час встановлення системи.

Загальна мета Частини 5 у забезпеченні тимчасової області, яка має у собі свідомо справного набору інструментів, які можуть бути ізольованими від хостової системи. Використовуючи **chroot,** команди у решті частинах будуть вміщатися у цьому інструментарії, забезпечуючи чистоту, вільну від проблем побудову кінцевої системи LFS. Процес побудови був створений для того щоб мінімізувати ризик для нових читачів і забезпечити навчання в один час.

> Увага
>
> Перед тим як продовжувати, будьте впевнені в назві платформи, на якій ви працюєте, яка часто називається платформова трійця. Найпростіший шлях визначення назви платформи це виконати скрипт config.gues, який постачається з кодом у багатьох пакетах. Розпакуйте дерево коду Binutils і виконайте скрипт командою: **./config.gues** і запам'ятайте її вивід. Для прикладу, для сучасних 32-бітних процесорів Intel вивід буде виглядати приміром i686-pc-linux-gnu.
>
> Також необхідно знати назву динамічного компонувальника платформи, який часто називається динамічним завантажувачем (не плутати з стандартним компонувальником **ld,** який є частиною Binutils). Динамічний компонувальник який постачається разом з Glibc знаходить і завантажує динамічні (англ. shared) бібліотеки необхідні для програми, підготовлює програми до виконання і виконує її. Ім'я динамічного компонувальника для 32-бітних процесорів Intel буде ld-linux.so.2. Безпомилковий шлях визначення ім'я динамічного компонувальника, полягає у перевірці будь-якого бінарного файлу від хостової системи, виконуючи команду: **readelf -l <назва бінарника> | grep interpreter** і запам'ятовуючи її вивід. Авторитетний довідник, який покриває усі платформи є у файлі shlib-versions в кореневому каталозі дерева коду Glibc.

Деякі основні технічні пункти які роз'яснюють як працює метод побудови Частини 5:

- Малим налаштуванням ім'я платформи, змінюючи поле “постачальника” платформової трійці у змінній LFS_TGT, забезпечує першу побудову пакетів Binutils і GCC створює сумісний крос компонувальник і крос компілятор. Замість того, щоб збирати бінарні для іншої архітектури, крос

компонувальник і крос компілятор будуть створювати бінарні файли, які сумісні з поточним апаратним забезпеченням.

- Тимчасові бібліотеки є крос скомпільовані. Тому, що крос компілятор за своєю природою не може опиратися ні нащо з його хост системи, цей метод відкидає потенційне забруднення вихідної системи, зменшуючи шанс включення заголовних файлів і бібліотек хостової системи у нові інструменти. Крос компіляція також дозволяє можливість побудови обох 32-бітних і 64-бітних бібліотек на сумісне з 64-бітною архітектурою апаратне забезпечення.

- Обережна маніпуляція з вихідним кодом GCC вказує компілятору який динамічний компонувальник буде використовуватися у системі.

Binutils встановлюються першими, оскільки скрипт configure на обох пакетах GCC і Glibc виконує деякі функціональні тести на асемблері і компонувальнику для визначення які функції програмного забезпечення ввімкнути чи вимкнути. Це є більш важливим ніж спочатку може здатись. Неправильно сконфігуровані GCC і Glibc можуть видати дещо несправний інструментарій, де вплив такої помилки можу не виявлятися поки не досягнутий кінець побудови системи. Провали тестових наборів, зазвичай, висвітлять ці помилки перед тим, як повинна бути виконана велика кількість додаткової роботи.

Binutils встановлює свій асемблерний компілятор і компонувальник у двох місцях, /tools/bin і /tools/$LFS?TGT/bin. Інструменти в одній директорії є жорсткими посиланнями на такі ж у іншій. Важливим аспектом компонувальника є його порядок пошуку бібліотек. Детальну інформацію можна отримати від **ld,** виконуючи цю команду з прапорцем -—verbose. Для прикладу, команда **ld —verbose | grep SEARCH** проілюструє поточні шляхи пошуку і їхню послідовність. Воно показує які файли є скомпоновані програмою **ld** компілюючи штучну програму і використовуючи параметр —verbose. Для прикладу, **gcc dummy.c -Wl, --verbose 2>&1 | grep succeeded,** покажу усі успішно відкриті файли під час компонування.

Настпний пакет, який слід встановити, буде GCC. Як приклад того, що може бути побаченим під час виконання скрипту **configure**:

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

Це є важливо через причини, сказані вище. Це також демонструє, що скрипт конфігурації GCC не шукає директорії, зазначені у PATH, щоб знайти які інструменти використовувати. Однак, під час самої операції з gcc, ті самі шляхи пошуку не використовуються обов'язково. Щоб визначити, який компонувальник стандартно використовує gcc, виконайте команду: **gcc -print-prog-name=ld**.

Детальну інформацію можна отримати від gcc з параметром -v поки компілюється макетна програма. Для прикладу, **gcc -v dummy.c** покаже вам детальну інформацію про препроцесор, компілювання і стадію асемблювання, включаючи шляхи пошуку заголовкових файлів gcc і їхній порядок.

Наступними встановлюються чисті заголовкові файли API Лінукс (application programming interface — інтерфейс програмування додатків). Це дозволяє стандартній бібліотеці C (Glibc) мати доступ до функцій, які поставляє ядро Лінукс.

Наступним пакунком буде Glibc. Найбільш необхідними для побудови Glibc є компілятор, бінарний набір інструментів, і заголовкові файли ядра. Компілятор не є зазвичай проблемою, через те що Glibc завжди використовує параметр —host на дається до його скрипту конфігурації, у нашому випадку це **i686-lfs-linux-gnu-gcc.** З бінарним набором інструментів і заголовкових файлів ядра можуть бути дещо більші проблеми. Тому, не ризикуйте і використовуйте доступні конфігураційні параметри щоб дотриматися правильних параметрів. Після запуску скрипту **configure**, перевірте важливий контент фалу config.make у директорії

glibc-build. Зауважте, що використання CC="i686-lfs-gnu-gcc" для контролю використання інструментів і використовуйте прапорці -nostdinc і -isystem для контролю шляхів пошуку компілятора. Ці пункти освітлюють важливі аспекти пакету Glibc — він дуже самостійний у терміні його механізму будування і яка, в загальному, не залежить від параметрів набору інструментів.

Під час наступного проходження Binutils, ми можемо використовувати конфігураційний параметр —with-lib-path для контролю шляхів пошуку бібліотек **ld**.

Для другого збирання GCC, його код також буде необхідно модифікувати щоб вказати GCC використовувати новий динамічний компонувальник. Провал у цьому спричинить що програми GCC будуть мати ім'я динамічного компонувальника з директорії /lib хостової системи, що спричинить недосяжність цілі незацлежності від хостової системи. Від цієї точки надалі, основний інструментарій є повністю самостійним. Залишок пакетів з Частини 5 побудує нову Glibc у директорію /tools.

На вході в середовище роботи за допомогою команди **chroot** у Хастині 6, першим головним пакунком для встановлення є Glibc, відносно його самостійної манери згаданої вище. Як тільки Glibc буде встановленою в /usr, ми зробимо швидкий перехід з звичайного набору інструментів, і розпочнемо процес побудови решти цільової системи LFS.

5.3. Загальні інструкції компілювання

Під час побудови пакетів, є кілька припущень зроблених в інструкціях:

- Декілька з пакетів будуть пропатчені перед компіляцією, але тільки коли патч є необхідним для запобігання проблеми. Патч є часто потрібним у убох - цій і наступній частині, але інколи тільки в одній або в іншій. Тому, не будьте стурбованими якщо інструкції для завантаження патчу відсутні. Попереджувачі повідомлення про зсув можуть також зустрічатися, коли ви застосовуєте патч. Не хвилюйтеся про ці повідомлення, так як патч убув успішно застосований.

- Під час компіляції більшості пакетів, будуть деякі попереджень які прокручуються на екрані. Це є нормально і може безпечно бути проігноровано. Ці попередження є повідомлення про застарілі, але не помилкові використання синтаксису C чи C++. Стандарт мови C змінюється достатньо часто і деякі пакети досі використовують старий стандарт. Це не є проблемою, але показує ці попередження.

- Перевірте в останній раз, чи змінна LFS встановлена правильно:

```
echo $LFS
```

  Впевніться, що вивід показує шлях до точки монтування розділу LFS, який має значення /mnt/lfs (у нашому випадку).

- Нарешті, два останні пункти повинні бути наголошеними:

Важливо

Інструкції побудови передбачає, що використовується командний інтерпретатор **bash**.

Важливо

Ще раз підкреслимо процес побудови:

1. Помістіть усі архіви пакетів і патчі у диреторію, яка буде доступна з робочого середовища після виконання команди **chroot**, таких як /mnt/lfs/sources. Не поміщайте їх у папку /mnt/lfs/tools.

2. Зайдіть в директорію джерел коду (архіви пакетів).

3. Для кожного пакету:

а) Виуористовуючи програму **tar**, витягніть пакети, які необхідно зібрати. У Главі 5, впевніться що ви зайшли під користувачем lfs, перед тим як розпаковувати пакети.

б) Зайдіть в директорію створену коли пакет було розархівовано.

в) Дійте згідно інструкцій вказаним у книзі для побудови пакету.

г) Поверніться до директорію джерел коду.

д) Видаліть розархівоване дерево джерел коду і будь-які директорії <назва пакету>-build, які були створені під час процесу побудови, якщо тільки інструкції твердили інше.

5.4. Binutils-2.22 — прохід 1

Пакет Binutils вміщає в собі компонувальник, компілятор мови Асемблер, і інші інструменти для обробки об'єктних файлів.

Приблизний час побудови: 1 SBU
Необхідний дисковий простір: 391 Мбайт

5.4.1. Встановлення крос Binutils

> Увага
>
> Поверніться і перечитайте пункти в попередній секції. Розуміння цих виділених пунктів є важливо, і вбереже вас від майбутніх проблем.

Дуже важливо, що Binutils буде першим скомпільованим пакетом тому, що Glibc і GCC роблять декілька тестів на доступному компонувальнику і асемблерному компіляторі, щоб визначити їхні власні увімкнені функції.

Застосуйте патч для запобігання помилок будування, при використанні методів оптимізації компілятора

```
patch -Np1 -i ../binutils-2.22-build_fix-1.patch
```

Документація рекомендує будувати Binutils ззовні директорії дерева вихідного коду — у спеціальній директорії компілювання:

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

> Увага
>
> У відповідності до одиниці SBU, яка зустрічається протягом читання усієї книги, для буду-якого використання, виміряйте час, який необхідний для побудови цього пакету від конфігурації, аж до першого встановлення. Щоб зробити це легшим, виконуйте усі три команди, обгорнувши їх за допомогою команди **time**: **time { ./configure ... && make && make install; }**.
>
> Увага
>
> Приблизний значення SBU побудови і необхідний дисковий простір у Частині 5 не включає масиви даних наборів тестування.

Тепер, підготуйте Binutils до компіляції:

```
../binutils-2.22/configure
    --prefix=/tools
    --with-sysroot=$LFS
    --with-lib-path=/tools/lib
    --target=$LFS_TGT
    --disable-nls
    --disable-werror
```

Значення опцій конфігурування:

`--prefix=/tools`

Ця опція вказує скрипту конфігурації підготувати до Binutils встановлення у директорії /tools.

--with-sysroot=$LFS

Для крос компілювання, це вказує системі побудови шукати під шляхом $LFS бібліотеки системи, якщо це необхідно.

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

Це встановлює шлях до пошуку бібліотек, які компонувальник буде сконфігурований використовувати.

--target=$LFS_TGT

Через те, що опис машини у змінній LFS_TGT дещо відрізняється від того, що повернув скрипт config.gues, це скаже скрипту configure змінити систему побудови Binutils для побудови крос компонувальника.

--disable-nls

Це вимикає інтернаціоналізацію, так як i18n не потрібна для тимчасового інструментарію.

--disable-werror

Цей запобігає зупинку побудови від настання події надходження попереджувальних повідомлень від компілятора хосту.

Продовжуйте компілюванням пакету:

```
make
```

Компіляція закінчилась. Зазвичай ми зараз почали б виконувати набір тестів, але це є рання стадія тому структура наборів тестів (Tcl. Expect і DejaGNU) ще не є на місці. Переваги від виконання тестів у цій точці є мінімальними, раз програми у цьому першому проходженні будуть найближчим часом замінені з другого.

Якщо будування відбувається на платформі x86_64, створіть посилання для забезпечення розумності інструментарію:

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

Встановіть пакет:

```
make install
```

Деталі цього пакету розміщуються у Секції 6.13.2, “Вміст Binutils”.

5.5. GCC-4.7.1 — прохід 1

Пакет GCC вміщує колекцію компіляторів GNU, які вміщують в собі компілятори мов C і C++.

Приблизний час побудови: 5.5 SBU
Необхідний дисковий простір: 1.4 Гбайти

5.5.1. Встановлення крос GCC

Зараз GCC вимагає пакети GMP, MPFR і MPC. Так як ці пакети можуть бути не включені у ваш дистрибутив, вони будуть побудовані разом з GCC. Розпакуйте кожен пакет у директорію дерева коду GCC і перейменуйте вихідні директорії так, щоб процедури побудови GCC використали їх автоматично.

> Увага
>
> Відбувається часте нерозуміння цієї частини. Процедура є такою ж як і в усіх інших частинах, як було прояснено раніше (Інструкції побудови пакету). Спочатку розпакуйте архів gcc з дерева усіх архівів і тоді перейдіть в створену директорію. Тільки тоді ви маєте продовжувати з інструкції, які розміщені дальше.

```
tar -Jxf ../mpfr-3.1.1.tar.xz
mv -v mpfr-3.1.1 mpfr
tar -Jxf ../gmp-5.0.5.tar.xz
mv -v gmp-5.0.5 gmp
tar -zxf ../mpc-1.0.tar.gz
mv -v mpc-1.0 mpc
```

Наступні команди змінять місце розміщення динамічного компонувальника який використовуватиме GCC, і вкаже на використання встановленого у директорії /tools. Це також виключає використання директорії */usr/include* з шляхів пошуку заголовкових файлів GCC. Команда:

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

Дана команда здається тяжкою до розуміння, давайте об'снемо деякі моменти. Спочатку ми знаходимо усі файли у директорії `gcc/config` які називаються `linux.h`, `linux64.h` або `sysv4.h`. Кожний знайдений файл, ми копіюємо але з додаванням суфіксу ".orig". Після цього перший sed-вираз підготовлює "tools" до кожного входження "/lib/ld", "/lib64/ld" чи "/lib32/ld", поки другий вираз замінює у коді входження рядка "/usr". Наступним, ми додамо наші define-твердження, які змінюють префікс початкових файлів за замовчуванням у кінець файлу. Зуважте, що передування "/" у "/tools/lib/" є необхідним. На кінець, ми використовуємо команду touch щоб оновити мітку часу копійованих файлів. Коли ми застосовуємо кон'юнкцію cp -u, це запобігає неочікуваним змінам оригінальних файлів у випадку коли команди неочікувано виконуються двічі.

GCC не виявляє коректно захист стеку, що спричиняє проблеми для побудови Glibc, отож поправте це виконуючи дану команду:

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

Документація GCC рекумендую побудову GCC ззовні кореневого дерева коду у спеціально створеній директорії:

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

Підготуйте пакет GCC до компіляції:

```
../gcc-4.7.1/configure \
    --target=$LFS_TGT \
    --prefix=/tools \
    --with-sysroot=$LFS \
    --with-newlib \
    --without-headers \
    --with-local-prefix=/tools \
    --with-native-system-header-dir=/tools/include \
    --disable-nls \
    --disable-shared \
    --disable-multilib \
    --disable-decimal-float \
    --disable-threads \
    --disable-libmudflap \
    --disable-libssp \
    --disable-libgomp \
    --disable-libquadmath \
    --enable-languages=c \
    --with-mpfr-include=$(pwd)/../gcc-4.7.1/mpfr/src \
    --with-mpfr-lib=$(pwd)/mpfr/src/.libs
```

Значення опцій конфігурації:

`--with-newlib`

Через те, що бібліотека C ще не доступна, це забезпечує, що константа inhibit_libc є визначеною коли будуємо libgcc. Це запобігає компіляції любого коду, який вимагає підтримку libc.

`--without-headers`

Коли створюється закінчений крос компілятор, GCC вимагає стандартні заголовкові файли, які сумісні з цільовою системою. Для нашої мети ці файли не будуть потрібними. Це запобігає шукання їх програмами GCC.

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

Локальний префікс являє собою місце в системі, де GCC буде робити пошук встановлених файлів. Значенням за замовчуванням є /usr/local. Встановлюючи його у /tools, допомагає тримати місця хостової системи, такі як /usr/local/ окремо від шляхів пошуку GCC.

`--with-native-system-header-dir=/tools/include`

За замовчуванням GCC робить пошук системних заголовкових файлв у директорії /usr/include. В зв'язку з вибором sysroot, це буде змінюватись в $LFS/usr/include. Однак заголовкові файли, які будуть встановленими у наступних двох секціях, будуть розміщеними у $LFS/tools/include. Цей

вибір забезпечує те, що gcc коректно знайде їх. У другому проходженні GCC, той самий вибір забезпечить те, що заголовкові файли з хостової системи не будуть знайденими.

--disable-shared

Цей вибір заставляє GCC компонувати його внутрішні бібліотеки статично. Ми робимо це, щоб запобігти можливим проблемам з хостовою системою.

--disable-decimal-float, --disable-threads, --disable-libmudflap, --disable-libssp, --disable-libgomp, --disable-libquadmath

Ці прапорці вимикають підтримку десяткову плаваючу кому, механізм потоків, бібліотеки libmudflap, libssp, libgomp і libquadmath відповідно. Ці функції провалять компілювання крос-компілятора і не є необхідними для завдання крос-компіляції тимчасової libc.

--disable-multilib

На архітектурах x86_64, LFS не підтримує багато бібліотечну конфігурацію. Цей вибір є нешкідливим для x86.

--eneble-languages=c

Ця опція вказує побудувати тільки компілятор мови C. Це є єдина мова яка зараз нам необхідна.

Скомпілюйте GCC виконавши команду:

```
make
```

Компіляція закінчена. У цій точці, Набір тестування може бути виконаним нормально, але, як згадано раніше, але структура тестового набору є не встановленою. Переваги виконання тестів у цій точці є мінімальні, так як дані програми будуть найближчим часом видалені.

Встановіть пакет:

```
make install
```

Використання *—disable-shared* означає, що файд *libgcc_eh.a* не є створеним і встановленим. Пакет Glibc залежить від цієї бібліотеки так як вона використовує *-lgcc-eh* з її системою побудови. Залежність може бути задоволена створенням посилання до *libgcc.a*, так як цей файл кінцевий вміст об'єктів зазвичай має нормальний вміст бібліотеки *libgcc_eh.a*

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

Деталі пакету ж в Секції 6.17.2, “Вміст GCC”.

5.6. Заголовкові файли API Linux-3.5.2

Заголовкові файли API Лінукс (в пакеті linux-3.5.2.tar.xz) надають можливість доступу бібліотеки Glibc до API Лінукс.

Приблизний час побудови: 0.1 SBU
Необхідний дисковий простір: 511 Мбайт

5.6.1. Встановлення заголовкових файлів API Лінукс

Ядро Лінукс має надавати Інтерфейс Програмування Додатків (API) для системної бібліотеки мови C (у LFS використовується Glibc). Це робиться шляхом витягування деяких заголовкових C файлів, які поміщені у архів вихідного коду Лінукс.

Впевніться, що немає ніяких залеглих файлів і залежностей які створилися під час попередньої активності:

```
make mrproper
```

Зараз протестуйте і витягніть заголовкові файли з усього коду. Вони є поміщеними в проміжній директорії і скопійованими у необхідне місце тому, що процес витягування видаляє будь-які файли, які існують в цільовій директорії.

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

Деталі цього пакету розміщені в Секції 6.7.2, “Склад заголовкових файлів API Лінукс”.

5.7. Glibc-2.16.0

Пакет Glibc вміщує головну бібліотеку C функцій. Ця бібліотека забезпечує базові підпрограми для виділення пам'яті, пошуку директорій, відкриття і закриття файлів і їх читання і запис , обробка рядків, пошук за шаблоном, арифметику і так далі.

Приблизний час побудови: 5.4 SBU
Необхідний дисковий простір: 554 Мбайт

5.7.1. Встановлення Glibc

У деяких випадках, особиво у LFS 7.1, заголовкові файли rpc були встановлені неправильно. Визначте, чи вони присутні в хостовій системі і встановіть якщо їх немає:

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

Виправте проблему поки будуєте Glibc-2.16.0 з GCC-4.7.1:

```
sed -i 's/ -lgcc_s//' Makeconfig
```

Документація до Glibc, рекомендує побудову даного пакету ззовні кореня дерева коду у спеціальній директорії:

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

Наступним чином, підготуйте пакет до компіляції:

```
../glibc-2.16.0/configure
     --prefix=/tools
     --host=$LFS_TGT
     --build=$(../glibc-2.16.0/scripts/config.guess)
     --disable-profile
     --enable-add-ons
     --enable-kernel=2.6.25
     --with-headers=/tools/include
     libc_cv_forced_unwind=yes
     libc_cv_ctors_header=yes
     libc_cv_c_cleanup=yes
```

Значення опцій конфігурування:

`--host=$LFS_TGT, --build=$(../glibc-2.16.0/scripts/config.guess )`

Ефект від комбінації цих параметрів у тому, що система побудови Glibc конфігурує себе до крос-компілювання, використовуючи крос-компонувальник і крос-компілятор у директорії */tools*.

`--disable-profile`

Це будує бібліотеки без профільної інформації. Опустіть ці опції, якщо профілювання тимчасових інструментів є необхідним.

`--enable-add-ons`

Це вказує Glibc використовувати додаток NPTL, як її бібліотеки потоків.

`--enable-kernel=2.6.25`

Це вказує компілювати бібліотеку Glibc з підтримкою для 2.6.25 і пізніших версій Лінукс. Шляхи використання старіших ядер не підтримуються.

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

Це вказує Glibc компілювати її на заголовкові файли, які були встановленими у директорію tools, щоб вона знала де саме розміщені функції ядра і може оптимізувати сама себе відповідно.

`libc_cv_forced_unwind=yes`

Компонувальник встановлений у Секції 5.5, “Binutils-2.22 прохід 1” є крос-скомпільованим і не може використовуватися поки Glibc не встановиться. Це означає, що конфігураційні тести для підтримки насильницького вимкнення - проваляться, так як вона покладається на працюючий компонувальник. Прапорець *libc_cv_forced_unwind=yes* встановлений для того щоб проінформувати **configure,** щоб підтримка примусового вимкнення є доступною без проходження тесту.

`libc_cv_c_cleanup=yes`

Аналогічно, ми вказали скрипту **configure** параметр `libc_cv_c_cleanup=yes`, отож тести будуть пропущені і обробка очищення C є увімкненою.

`libc_cv_ctors_headers=yes`

Знову, ми задаємо цей параметр скрипту **configure,** отож тест є пропущеним і конструктор gcc є сконфігурованим.

Під час цієї стадії можуть надійти наступні повідомлення:

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

Відсутність або несумісність програми msgmft є в загальному нешкідливим Програма **msgmft** є частиною пакету Gettext, який повинний надавати хостовий дистрибутив.

Скомпілюйте пакет:

```
make
```

Цей пакет поставляється з тестовим набором пограм, однак, він не можу бути виконаним тому, що ми це не маємо компілятора C++.

> Увага
>
> Тестовий набір вимагає встановлені дані локалізації, щоб успішно працювати. Дані локалізації забезпечують систему інформацією, таких речей як дата, час і формати валют, які система приймає і видає своїми утилітами. Якщо тестові набори не були виконаними в цій частині (у відповідності до рекомендацій), немає потреби встановлювати дані локалізації зараз. Відповідні локалізації будуть встановленими у наступних частинах книги. Для того, щоб все одно встановити локалізації Glibc, використовуйте інструкції з Частини 6.9, “Glibc-2.16.0”.

Встановіть пакет:

```
make install
```

Застереження

У цій точці, вам необхідно зупинитись і впевнитись, що базові функції (компілювання і компонування) нового інструментарію працюють як очікується. Для того щоб виконати розумну перевірку, виконайте наступні команди:

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

Якщо все працює правильно, не повинно бути ніяких помилок, і вивід останньої команди буде у формі:

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

Зазначте, що /tools/lib або /tools/lib64 для 64-бітних машин з'являється як префікс до динамічного компонувальника.

Якщо вивід не показує те, що вказано вище, або виводу не було взагалі - це означає, що щось пішло не правильно. Ця проблема повинна бути вирішеною перед тим як продовжувати процес.

Якщо все добре, очистіть директорію від тестових файлів:

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

Увага

Будування Binutils у наступній секції, буде виконувати роль і як додаткової перевірки, що інструментарій був побудованим правильно. Якщо Binutils не побудуються, це є ознакою того, що щось з попередніми встановленими Binutils, GCC чи Glibc пішло неправильно.

Деталі цього пакету розміщуються у Секції 6.9.4, “Вміст Glibc”.

5.8. Binutils-2.22 — прохід 2

Пакет Binutils містить у собі компонувальник, асемблерний компілятор і інші інструменти для обробки об'єктних файлів.

Приблизний час побудови: 1.1 SBU
Необхідний дисковий простір: 407 Мбайт

5.8.1. Встановлення Binutils

Застосуйте патч для запобігання помилок побудови при використанні механізмів оптимізації компілятора:

```
patch -Np1 -i ../binutils-2.22-build_fix-1.patch
```

Створіть окрему директорію для побудови:

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

Підготуйте Binutils до компіляції:

```
CC=$LFS_TGT-gcc \
AR=$LFS_TGT-ar \
RANLIB=$LFS_TGT-ranlib \
../binutils-2.22/configure \
    --prefix=/tools \
    --disable-nls \
    --with-lib-path=/tools/lib
```

Значення нових опцій компілювання:

CC=$LFS_TGT-gcc AR=$LFS_TGT-ar RANLIB=$LFS_TGT-ranlib

Через те, що це є побудова Binutils, яка буде розміщуватися в системі, встановлення цих параметрів забезпечує те, що система побудови використає крос-компілятор і асоційовані з ним інструменти замість того, щоб використовувати їх з хостої системи.

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

Це вказує скрипту configure задати шлях пошуку бібліотек під час компілювання Binutils, який передається як `/tools/lib`. Це запобігає пошук бібліотек компонувальника у директоріях хостової системи.

Скомпілюйте пакет:

```
make
```

Встановіть пакет:

```
make install
```

Зараз, підготуйте компонувальник до фази перевстановлення у наступній частині:

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

Значення параметрів:

-C ld clean

Це вказує прорамі make видаляти усі скомпільовані файли у піддиректоріях ld.

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

Ця опція перебудовує усе у піддиректорії ld. Вказуючи змінну файлу Makefile у команді, дозволяє нам змінити значення за замочуванням тимчасових інструментів і вказати його на належні фінальні шляхи. Значення цієї змінної визначає шлях пошуку бібліотек компонувальника за замовчуванням. Ці підготування використовуються у наступній частині.

Деталі цього пакету розміщені в секції 6.13.2, “Вміст Binutils”.

5.9. GCC-4.7.1 — прохід 2

Пакет GCC вміщає колекцію компіляторів GNU, що включає компілятори мов C і C++.

Приблизний час побудови: 7.1 SBU
Необхідний дисковий простір: 1.8 Гбайт

5.9.1. Встановлення GCC

Наша перша побудова GCC встановила декілька внутрішніх заголовкових файлів системи. Зазвичай один з них, `limits.h` в свою чергу підключає відповідний системний заголовковий файл `limits.h`, у цьому випадку, `/tools/include/limits.h`. Однак, під час першої побудови gcc файл `/tools/include/limits.h` не існує, отож внутрішній заголовковий файл, що GCC встановлює є не повним, автономним файлом і не підключає розширені можливості системних заголовкових файлів. Це було адекватним для побудови тимчасової бібліотеки Glibc, але ця побудова GCC вимагає повні внутрішні заголовкові файли. Створюючи повні версії внутрішніх заголовків, використовуючи команду, яка є ідентичною до тої, що система побудови GCC робить в нормальних обставинах:

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

Для систем архітектури x86, початкова побудова GCC використовує прапорець компілятора `-fomit-frame-pointer`. Інша побудова пропускає цей прапорець за замовчуванням, і ціллю повинне бути створити компілятор, який є таким самим які початковий. Задійте наступні **sed**-команди для того щоб змусити побудову використовувати цей прапорець:

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

Ще раз, змініть звичайне місце знаходження динамічного компонувальника GCC на той, що встановлений у `/tools`.

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

Так як в першій побудові GCC, вимагаються пакети GMP, MPFR і MPC. Розпакуйте архіви і перемістіть їх в необхідну імена директорій:

```
tar -Jxf ../mpfr-3.1.1.tar.xz
mv -v mpfr-3.1.1 mpfr
tar -Jxf ../gmp-5.0.5.tar.xz
mv -v gmp-5.0.5 gmp
tar -zxf ../mpc-1.0.tar.gz
mv -v mpc-1.0 mpc
```

Створіть окрему директорію для побудови:

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

Перед тим як розпочати побудову GCC, запам'ятайте видалити усі змінні середовища, які зазвичай скасовують оптимізаційні прапорці.

Тепер, підготуйте GCC до компіляції:

```
CC=$LFS_TGT-gcc \
AR=$LFS_TGT-ar \
RANLIB=$LFS_TGT-ranlib \
../gcc-4.7.1/configure \
    --prefix=/tools \
    --with-local-prefix=/tools \
    --with-native-system-header-dir=/tools/include \
    --enable-clocale=gnu \
    --enable-shared \
    --enable-threads=posix \
    --enable-__cxa_atexit \
    --enable-languages=c,c++ \
    --disable-libstdcxx-pch \
    --disable-multilib \
    --disable-bootstrap \
    --disable-libgomp \
    --with-mpfr-include=$(pwd)/../gcc-4.7.1/mpfr/src \
    --with-mpfr-lib=$(pwd)/mpfr/src/.libs
```

Значення нових опцій:

`--enable-clocale=gnu`

Ця опція забезпечує коректне встановлення моделі локалізації для бібліотек C++ у всіх обставинах. Якщо скрипт конфігурації знаходить встановлену локалізацію *de_DE*, він встановить відповідну модель локалізації gnu. Однак, якщо локалізація *de_DE* не встановлена, є ризик побудови ABI (Application Binary Interface — бінарний програмний інтерфейс) несумісного з бібліотеками C++ тому, що можу бути вибрана не коректна загальна модель локалізації.

`--enable-threads=posix`

Це вмикає механізм обробки виключень C++ для багатопотокового коду.

`--enable-__cxa_atexit`

Ця опція дозволяє використовувати `__cxa_atexit`, замість `atexit`, щоб реєструвати деструктори C++ для локальних статичних і глобальних об'єктів. Ця опція є істотною для повної сумісності з стандартом обробленням деструкторів. Це також впливає на C++ ABI, і, внаслідок цього, впливає на розділені бібліотеки C++ і програми, що взаємодіють між іншими дистрибутивами Лінукс.

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

Ця опція запевняє побудову компіляторів мов програмування C і C++.

`--disable-libstdcxx-pch`

Ця опця запобігає будуванню попередньо скомпільованих заголовкових файлів (PCH) для libstdc++.

Це займає багато місця, і ми не використовуємо їх.

`--disable-bootstrap`

Для рідних збірок GCC, за замовчуванням є початковими (“bootstrap”). Він не просто компілює GCC, а й відповідає його декілька разів. Він використовує програми з першого проходу, щоб скомпілювати себе в другому, і тоді знову в третій раз. Друга і третя ітерація порівнюються, щоб впевнитись чи може він відмінно відтворювати себе. Це також доказує, що він був побудований коректно. Однак, метод побудови LFS повинен забезпечувати солідний компілятор без потреби у постійному бутстрапінгу.

Скомпілюйте пакет:

```
make
```

Інсталюйте його:

```
make install
```

Фінальною дією, буде створення посилань. Багато програм і скриптів виконуються **cc** замість **gcc**, щоб зберегти програми стандартними і отже можливими до використання на усіх типах UNIX-систем де компілятор мови C GNU не завжди встановлений. Використовуючи **cc** залишає системному адміністратору свободу вирішувати який встановлювати компілятор мови C:

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

Увага

Тестовий набір вимагає встановлені дані локалізації, щоб успішно працювати. Дані локалізації забезпечують систему інформацією, таких речей як дата, час і формати валют, які система приймає і видає своїми утилітами. Якщо тестові набори не були виконаними в цій частині (у відповідності до рекомендацій), немає потреби встановлювати дані локалізації зараз. Відповідні локалізації будуть встановленими у наступних частинах книги. Для того, щоб все одно встановити локалізації Glibc, використовуйте інструкції з Частини 6.9, “Glibc-2.16.0”.

Встановіть пакет:

```
make install
```

Застереження

У цій точці, вам необхідно зупинитись і впевнитись, що базові функції (компілювання і компонування) нового інструментарію працюють як очікується. Для того щоб виконати розумну перевірку, виконайте наступні команди:

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

Якщо все працює правильно, не повинно бути ніяких помилок, і вивід останньої команди буде у формі:

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

Зазначте, що /tools/lib або /tools/lib64 для 64-бітних машин з'явиться як префікс динамічного компонувальника.

Якщо вивід не подібний до показаного вище, або взагалі немає ніякого виводу на екран, це означає, що щось пішло не так. Дослідіть і простежте кроки, щоб дізнатись де знаходиться проблема і виправити її. Ця проблема повинна бути вирішеною перед тим, як продовжувати далі. Спочатку, виконайте повну перевірку знову, використовуючи **gcc** замість **cc**. Якщо все запрацювало, тоді зникло посилання /tools/bin/cc. Встановіть посилання як було показано вище. Після цього, впевніться, що змінна середовища PATH є коректно встановленою. Це можна перевірити виконуючи команду **echo $PATH** і виявити, чи шлях /tools/bin/ є спочатку списку. Якщо змінна PATH є неправильно встановленою, це може означати, що ви не ввійшли в ситему як користувач lfs, або щось пішло неправильно у Секції 4.4, “Підготовлення робочого середовища”.

Якщо все добре, видаліть тестові файли:

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

Деталі цього пакету розміщені у Секції 6.17.2, “Вміст GCC”.

5.10. Tcl-8.5.12

Пакет Tcl вміщує Мову Коммандного Інструментарію (Tool Command language).

Приблизний час побудови: 0.4 SBU
Необхідний дисковий простір: 33 Мбайти

5.10.1 Встановлення Tcl

Цей пакет і наступні три (Expect, DejaGNU і Check) встановлюються для підтримки наборів тестування для GCC і Binutils і інших пакетів. Встановлюючи чотири пакети для цілей тестування може здаватись марним, але для цього є свої причини: нам необхідно взнати чи усі інструменти працюють правильно. Навіть якщо набори тестування не будуть виконуватися у цій частині (вони не обов'язкові), ці пакети вимагаються у Главі 6.

Підготуйте Tcl до компіляції:

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

Зберіть пакет:

```
make
```

Зараз компілювання закінчено. Як і було обговорено раніше, виконання тестових пакетів у цій частині не є обов'язковим для тимчасових інструментів. Для виконання тестового пакету Tcl, виконайте наступну команду:

```
TZ=UTC make test
```

Тестовий пакет Tcl може видати декілька помилок від деяких особливостях хосту, які ще не є зрозумілими. Однак, тестові помилки тут не є сюрпризом, і не є критичними. Параметер TZ=UTS встановлює часовий пояс у Coordinated Universal Time (UTS — універсальний координований час), який ще називається Greenwich Mean Time (GMT), але тільки на час виконання тестів. Це запевняє, що тести часу виконуються правильно. Деталі змінної середовища TZ розміщені у Частині 7.

Встановіть пакет:

```
make install
```

Зробіть інстальовану бібліотеку доступною до запису щоб налагоджуванні символи могли бути вилучені у майбутньому.

```
chmod -v u+w /tools/lib/libtcl8.5.so
```

Встановіть заголовкові файли Tcl. Наступний пакет — Expect, вимагає їх для побудови.

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

Тепер зробіть необхідне символічне посилання:

```
ln -sv tclsh8.5 /tools/bin/tclsh
```

5.10.2 Вміст пакету Tcl

Встановлені програми: tclsh (посилання на tclsh8.5) і tclsh8.5

Встановлені бібліотеки: libtcl8.5.so, libtclstub8.5.a

Короткий опис

tclsh8.5	Командний інтерпретатор tcl
tclsh	Посилання на tclsh8.5
`libtccl8.5.so`	Бібліотека Tcl
`libtclstub8.5.a`	Бібліотека Tcl Stub

5.11. Expect-5.45

Пакет Expect вміщає програми для підтримки скриптових діалогів з іншими інтерактивними програмами.

Приблизний час побудови: 0.1 SBU
Необхідний дисковий простір: 4.4 Мбайт

5.11.1. Встановлення Expect

Для початку, змусьте скрипт configure використовувати /bin/stty замість /usr/local/bin/stty, він може знайти його на хостовій машині. Це забезпечить, що наші тестові набори будуть адекватними для фінальної побудови нашого інструментарію.

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

Тепер підготуйте Expect для компіляції:

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

Значчення конфігураційних опцій:

`--with-tcl=/tools/lib`

Це запевняє, що конфігураційний скрипт знайде встановлений Tcl в місці тимчасових інструментів замість знаходження такої ж програми на системі хосту.

`--with-tclinclude=/tools/include`

Це неоднозначно вказує Expect, де саме у тимчасових інструментах шукати встановлений Tcl, замість використання такого у хостовій системі. Використання цієї опції запобігає умови, при яких configure провалить свою роботу тому, що він не зможе автоматично встановити місце знаходження заголовкових файлів Tcl.

Побудуйте пакет:

```
make
```

Компілювання зараз завершене. Як було обговорено раніше, виконання тестових наборів не є хобов'язковим для тимчасових інструментів у цій частині. Для того щоб все одно виконати тестові набори Expect, виконайте наступну команду:

```
make test
```

Запам'ятайте, що тестові набори Expect можуть провалитися від деяких умов, які не є під нашим контролем. Однак, помилки тестів не є сюрпризом і не є критичними.

Встановіть пакет:

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

Значення параметрів make:

`SCRIPT=""`

Це запобігає встановленню деяких скриптів Expect, які не є потрібними.

5.11.2. Вміст пакету Expect

Встановлена програма: expect

Встановлена бібліотека: libexpect-5.45.a

Короткий опис

expect	З'єднується з іншими інтерактивними програмами у відповідності до скрипту.
`libexpect-5.45.a`	Вміщує функції, які дозволяють використовувати Expect як розширення Tcl, або прямо з C чи C++ (без Tcl).

5.12. DejaGNU-1.5

Пакет DejaGNU вміщує інструменти для тестування інших програм.

Приблизний час побудови: менше ніж 0.1 SBU
Необхідний дисковий простір: 4.1 Мбайт

5.12.1. Встановлення DejaGNU

Підготовка DejaGNU до компіляції:

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

Побудуйте і встановіть пакет:

```
make install
```

Для тестування результатів:

```
make check
```

5.12.2. Вміст DejaGNU

Встановлена програма: runtest

Короткий опис

runtest скрипт-обгортка, що встановлює розміщення правильної командної оболонки **expect** і тоді виконує DejaGNU.

5.13. Check-0.9.8

Check є одиницею тестувального інструментарію для C.

Приблизний час побудови: 0.1 SBU
Необхідний дисковий простір: 6.9 Мбайт

5.13.1. Встановлення Check

Підготуйте Check до встановлення:

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

Побуйте пакет:

```
make
```

Компілювання завершено. Як було обговорено раніше, виконання тестових програм не є обов'язковим для тимчасових інструментів у цій частині. Для того щоб все таки виконати їх, виконайте наступну команду:

```
make check
```

Зверніть увагу, що тестовий набір Check може зайняти відносно великий час (до 4 SBU).

Встановіть пакет

```
make install
```

5.13.2. Зміст Check

Встановлені бібліотеки: `libcheck.{a,so}`

Короткий опис

`libcheck.{a,so}`	Вміщує функції, які дозволяють програмі Check бути викликаною з іншої програми тестування.

5.14. Ncurses-5.9

Пакет Ncurses вміщує бібліотеки для обробки символів екрану незалежних від терміналу.

Приблизний час побудови: 0.5 SBU
Необхідний дисковий простір: 35 Мбайт

5.14.1. Встановлення Ncurses

Підготуйте Ncurses до компілювання:

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

Значення опцій конфігурування:

`--without-ada`

Цей параметр запевняє, що Ncurses не побудує підтримку мови Ada, який може бути присутнім на хості, але не буде доступним, коли ми використаємо команду chroot.

`--enable-overwrite`

Це вказує пакету Ncurses встановити його заголовкові файли у директорію `/tools/include`, замість того щоб встановлювати їх у `/tools/include/ncurses`, для забезпечення того, що пакети зможуть успішно знайти заголовкові файли Ncurses.

Скомпілюйте пакет

```
make
```

Цей пакет має тестовий набір, але він може бути виконаним тільки після того, як Ncurses буде встановлено. Тест розміщується у директорії `test/`. Прочитайте файл README (англ., дослівно — “прочитай мене”) у цій директорії для додаткової інформації.

Встановіть пакет:

```
make install
```

Деталі цього пакету розміщені у Секції 6.21.2 “Вміст Ncurses”.

5.15. Bash-4.2

Пакет Bash вміщує Bourne-Again Shell (командний інтерпретатор).

Приблизний час побудови: 0.4 SBU
Необхідний дисковий простір: 48 Мбайт

5.15.1. Встановлення Bash

Спочатку, застосуйте наступний патч, щоб виправити деякі помилки, які були вказані потоком користувачів.

```
patch -Np1 -i ../bash-4.2-fixes-8.patch
```

Підготуйте Bash до компіляції:

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

Значення опцій конфігурування:

`--without-bash-malloc`

Ця опція відключає використання функції Bash - виділення пам'яті, яка, як відомо, спричиняє помилки роботи з сегментами. Виключаючи цю опцію, Bash буде використовувати функцію `malloc` з бібліотеки Glibc, яка є стабільнішою.

Скомпілюйте пакет:

```
make
```

Компілювання вже закінчилось. Як було обговорено раніше, виконання тестів не є обов'язковим на даній стадії будування тимчасових інструментів в цій частині. Для того, щоб все таки виконати цей набір тестів, виконайте наступну команду:

```
make test
```

Встановіть пакет:

```
make install
```

Зробіть посилання для програм, котрі використовують sh для виклику командного інтерпретатора:

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

Деталі цього пакету розміщені в Секції 6.33.2, “Вміст пакету Bash”.

5.16. Bzip-1.0.6

Пакет Bzip вміщує у собі програми для компресії і декомпресії файлів. Стиснення текстових файлів за допомогою програми bzip2 дає набагато більший процент стиснення ніж традиційний gzip.

Приблизний час побудови: менше ніж 0.1 SBU
Необхідний дисковий простір: 5.7 Мбайт

5.16.1 Встановлення Bzip2

Пакет Bzip2 не має у собі скрипт configure. Скомпілюйте його і протестуйте виконуючи:

```
make
```

Встановіть пакет:

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

Деталі цього пакету розміщені у секції 6.19.2, “Вміст Bzip2”

5.17. Coreutils-8.19

Пакет Coreutils поставляє утиліти для показування і встановлення базових системних характеристик.

Приблизний час побудови: 0.7 SBU
Необхідний дисковий простір: 126 Мбайт

5.17.1 Встановлення Coreutils

Підготування до компіляції:

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

Значення опцій конфігурування:

`--enable-install-program=hostname`

Ця опція вказує на необхідність побудови бінарного файлу **hostname** — це є вимкненим за замовчуванням, але вона вимагається для тестового пакету Perl.

Скомпілюйте пакет:

```
make
```

Компілювання є завершеним. Як було сказано раніше, виконувати тестовий пакет не є обов'язковим для тимчасових інструментів у цій частині книги. Для того, щоб все одно виконати тест, виконайте наступну команду:

```
make RUN_EXPENSIVE_TEST=yes check
```

Праметер `RUN_EXPENSIVE_TEST=yes`, вказує тестовому набору виконати додаткові тести, що є дуже витратними (у термінах живлення процесора і використання пам'яті) на деяких платформах, але у загальному не являється проблемою для Лінукс.

Встановіть пакет:

```
make install
```

Деталі цього пакету розміщуються у Секції 6.26.2, “Вміст Coreutils”

5.18. Diffutils-3.2

Пакет Diffutils вміщує програми, я кі показують різницю між файлами або директоріями.

Приблизний час побудови: 0.2 SBU
Необхідний дисковий простір: 8.5 Мбайт

5.18.1. Встановлення Diffutils

Виправте несумісність між цим пакетом і Glibc-2.16.0:

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

Підготуйте Diffutils до компіляції:

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

Встановіть пакет:

```
make
```

Компілювання завершено. Як і було обговорено раніше, виконання тестових наборів не є обов'язковим у цій частині книги. Але щоб його все таки виконати, використовуйте команду:

```
make check
```

Встановіть пакет:

```
make install
```

Деталі цього пакету розміщені у Секції 6.4.2, “Вміст Diffutils”.

5.19. File-5.11

Пакет File вміщує в собі програму для визначення типу даного файлу або файлів.

Приблизний час побудови: 0.1 SBU
Необхідний дисковий простір: 2.4 Мбайт

5.19.1. Встановлення File

Підготуйте пакет File до встановлення:

```
./configure —prefix=/tools
```

Скомпілюйте пакет:

```
make
```

Компілювання завершено. Як було сказано раніше, виконання тестів не є обов'язковим на даній стадії побудови. Якщо ви все таки хочете виконати тести, використовуйте наступну команду:

```
make check
```

Встановіть пакет:

```
make install
```

Деталі цього пакету розміщені у Секції 6.12.2, “Вміст File”.

5.20. Findutils-4.4.2

Пакет Findutils вміщує програми для пошуку файлів. Ці програми поставляються для рекурсивного пошуку крізь дерево директорій і для створення, підтримки і пошуку у базі даних (частіше швидшу, ніж рекурсивний пошук, але ненадійна, якщо база даних часто не оновлюється).

Приблизний час побудови: 0.2 SBU
Необхідний дисковий простір: 27 Мбайт

5.20.1. Встановлення Findutils

Підготуйте Findutils до встановлення:

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

Скомпілюйте пакет:

```
make
```

Зараз компілювання закінчено. Для того щоб виконати тест, використовуйте команду:

```
make check
```

Встановіть пакет:

```
make install
```

Деталі до цього пакету розміщені у Секції 6.42.2, “Вміст FindUtils”.

5.21. Gawk-4.0.1

Пакет Gawk вміщує в собі програми для маніпулювання текстовими файлами.

Приблизний час побудови: 0.2 SBU
Необхідний дисковий простір: 30 Мбайт

5.21.1. Встановлення Gawk

Підготуйте Gawk до компілювання:

```
./configure —prefix=/tools
```

Скомпілюйте пакет:

```
make
```

Для виконання тесту, використовуйте команду:

```
make check
```

Встановіть пакет:

```
make install
```

Деталі до цього пакету розміщені в Секції 6.41.2, “Вміст Gawk”.

5.22. Gettext-0.18.1.1

Пакет Gettext вміщує утиліти для інтернаціоналізації і локалізацій. Він дозволяє програмам бути скомпільованими з NLS (Native Language Support — підтримка рідної мови), дозволяючи їм виводити повідомлення на рідній мові користувача.

Приблизний час побудови: 0.6 SBU
необхідний дисковий простір: 101 Мбайт

5.22.1. Встановлення Gettext

Для нашого тимчасового набору інструментів, нам необхідно встановити тільки один бінарний файл від Gettext.

Виправте несумісність між цим пакетом і Glibc-2.16.0:

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

Підготуйте Gettext до компіляції:

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

Значення опцій конфігурування:

EMACS=”no”

Ця опція запобігає скрипт конфігурування від визначення місця встановлення файлів List Emacs`a, так як цей тест, як відомо, падає на деяких хостах.

--disable-shared

Нам не потрібно встановлювати будь-які поділювані бібліотеки (shared libraries) пакету Gettext цього разу, але немає необхідності будувати їх.

Скомпілюйте пакет:

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

Так як тільки один бінарник скомпільовано, це не можливо виконувати тест без компіляції підтримки бібліотек у пакеті Gettext. І не рекомендується виконувати тест на цій стадії.

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

Деталі цього пакету розміщуються у Секції 6.44.2, “Вміст Gettext”.

5.23. Grep-2.14

Пакет Grep містить програми для пошуку крізь файли.

Приблизний час побудови: 0.2 SBU
Необхідний дисковий простір: 21 Мбайт

5.23.1. Встановлення Grep

Підготуйте пакет до встановлення:

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

Скомпілюйте пакет:

```
make
```

Для тестування, виконайте команду:

```
make check
```

Встановіть пакет:

```
make install
```

Деталі цього пакету розміщені у Секції 6.31.2, “Вміст Grep”.

5.24. Gzip-1.5

Пакет Gzip вміщує програми для стискання і розтискання файлів.

Приблизний час побудови: 0.2 SBU
Необхідний дисковий простір: 10 Мбайт

5.24.1. Встановлення Gzip

Підготуйте пакет до компіляції:

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

Скомпілюйте пакет:

```
make
```

Для виконання тестів, використовуйте наступну команду (необов'язкових):

```
make check
```

Встановіть пакет:

```
make install
```

Деталі цього пакету розміщені у секції 6.49.2, “Вміст Gzip”.

5.25. M4-1.4.16

Пакет M4 вміщує макропроцесор.

Приблизний час побудови: 0.2 SBU
Необхідний дисковий простір: 16.6 Мбайт

5.25.1. Встановлення M4

Виправте несумісність між цим пакетом і Glibc-2.16.0.

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

Підготуйте M4 до компіляції:

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

Скомпілюйте пакет:

```
make
```

Для проведення тестів, виконайте команду (необов'язково):

```
make check
```

Встановіть пакет:

```
make install
```

Деталі про цей пакет ви можете знайти у Частині 6.28.2, “Вміст M4”.

5.26. Make-3.82

Пакет Make вміщує програму для компілювання пакетів.

Приблизний час побудови: 0.1 SBU
Необхідний дисковий простір: 11.2 Мбайт

5.26.1. Встановлення Make

Підготуйте Make до встановлення:

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

Для виконання тестів використовуйте команду (необов'язково):

```
make check
```

Встановіть пакет:

```
make install
```

Деталі про цей пакет ви можете знайти в Секції 6.54.2, “Вміст Make”.

5.27. Patch-2.6.1

Пакет Patch вміщує програму для модифікування чи створення файлів шляхом застосування файлу “патч” (“patch”), який зазвичай створюється програмою diff.

Приблизний час побудови: 0.1 SBU
Необхідний дисковий простір: 3.4 Мбайт

5.27.1. Встановлення Patch

Підготуйте Patch до компілювання:

```
./configure —prefix=/tools
```

Скомпілюйте пакет:

```
make
```

Для виконання тестів, використовуйте команду (необов'язково):

```
make check
```

Встановіть пакет

```
make install
```

Деталі цього пакету розміщені у Секції 6.56.2, “Вміст Patch”.

5.28. Perl-5.16.1

Пакет Perl поставляє інтерпретатор мови Perl (англ. Practical Extraction and Report Language — практична мова запитів і звітів).

Приблизний час побудови: 1.8 SBU
Необхідний дисковий простір: 237 Мбайт

5.28.1. Встановлення Perl

Спочатку, застосуйте наступний патч для адаптування деяких шляхів, встановлених прямо у коді, до бібліотеки C.

```
patch -Np1 -i ../perl-5.16.1-libc-2.patch
```

Підготуйте пакет до компілювання:

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

Побудуйте пакет:

```
make
```

Хоча пакет Perl поставляється з тестовим набором програм, буде краще зачекати поки він встановиться у наступній частині.

Зараз, тільки декілька утиліт і бібліотек необхідно встановити:

```
cp -v perl cpan/podlators/pod2man /tools/bin
mkdir -pv /tools/lib/perl5/5.16.1
cp -Rv lib/* /tools/lib/perl5/5.16.1
```

Деталі цього пакету розміщені у Секції 6.37.2, “Вміст Perl”.

5.29. Sed-4.2.1

Пакет Sed вміщує редактор потоку.

Приблизний час побудови: 0.1 SBU
Необхідний дисковий простір: 10.5 Мбайт

5.29.1. Встановлення Sed

Підготуйте пакет до встановлення:

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

Скомпілюйте пакет:

```
make
```

Для виконання тестів, застосуйте команду (необов'язково):

```
make check
```

Встановіть пакет:

```
make install
```

Деталі цього пакету розміщені у Секції 6.18.2, “Вміст Sed”.

5.30. Tar-1.26

Пакет Tar містить програму-архіватор.

Приблизний час побудови: 0.4 SBU
Необхідний дисковий простір: 20.6 Мбайт

5.30.1. Встановлення Tar

Виправте несумісність між цим пакетом і Glibc-2.16.0

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

Підготуйте пакет до компілювання:

```
make
```

Виконати тест, можна за допомогою команди (необов'язково):

```
make check
```

Встановіть пакет:

```
make install
```

Деталі цього пакету розміщені у Секції 6.59.2, “Вміст Tar”.

5.31. Texinfo-4.13a

Пакет Texinfo вміщує програми для читання, написання і перетворення сторінок info.

Приблизний час побудови: 0.2 SBU
Необхідний дисковий простір: 24 Мбайт

5.31.1. Встановлення Texinfo

Підготуйте пакет до встановлення:

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

Скомпілюйте пакет:

```
make
```

Для тестування, виконайте (необов'язково):

```
make check
```

Втановіть пакет:

```
make install
```

Деталі цього пакету розміщені в Секції 6.60.2, “Вміст Texinfo”.

5.32. Xz-5.0.4

Пакет Xz вміщує програми для компресії і декомпресії файлів. Він забезпечує можливості для форматів файлів lzma і новіших xz. Стискаючи текстові файли з xz забезпечує кращий процент компресії ніж традиційні програми gzip і bzip2.

Приблизний час побудови: 0.2 SBU
Необхідний дисковий простір: 16.3 Мбайт

5.32.1. Встановлення Xz-Utils

Підготуйте Xz до компіляції:

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

Скомпілюйте пакет

```
make
```

Для виконання тестів (необов'язково в цій частині), використовуйте команду:

```
make check
```

Встановіть пакет:

```
make install
```

Деталі цього пакету розміщуються у Секції 6.46.2, “Вміст Xz”.

5.33. Видаляння відлагоджувальної інформації

Кроки у цій секції є необов'язковими, але, якщо розділ LFS є малий, було б добре вивчити як видалити непотрібні пункти. Виконувані файли і бібліотеки побудовані так, що вміщують у собі 70 Мбайт непотрібні відлагоджувальні символи.

Усуньте ці символи, виконуючи:

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

Ці команди пропустять деякі файли, звітуючи, що вони не розпізнають їхній формат. Більшість з них є скриптами, а не бінарними файлами.

Попіклуйтеся, щоб не використовувати `--strip-unneeded` на бібліотеках. Статичні бібліотеки будуть знищеними і пакети інструментів потрібно буде побудувати знову.

Для збереження більшого простору, усуньте документацію:

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

У цій точці, ви повинні мати принаймні 850 Мбайт вільного простору у розділі LFS, який може бути використаний для побудови і встановлення пакету Glibc у наступній фазі. Якщо ви можете побудувати і встановити Glibc, ви можете побудувати і встановити решту пакетів.

5.34. Міняючи власників

> Увага
>
> Наступні команди книги повинні бути виконаними поки ви присутні в системі як користувач root і не як користувач lfs. Також, двічі перевірте, чи $LFS встановлена в кореневе середовище LFS.

В даний час, директорія `$LFS/tools` є у власництві користувача `lfs`, користувач який існує тільки на хостовій системі. Якщо тримати це як є, директорія `$LFS/tools`, власником файлів буде ID користувача без відповідного його профілю у системі. Це є небезпечним тому, що профіль користувача (system account), який буде створений пізніше, може отримати той самий ID (англ. IDentifier — ідентифікатор), буде власником директорії `$LFS/tools` і усіх файлів у ній, це піддає ці файли до можливих зловмисних маніпуляцій.

Щоб запобігти цій проблемі, ви можете добавити користувача `lfs` до нової системи LFS пізніше, коли будете створювати файл `/etc/passwd`, дбаючи про те, щоб прив'язати ті самі ID користувача і групи до які і в хостовій системі. Але краще, щоб ви змінили власника директорій `$LFS/tools` на користь користувача root виконуючи наступну команду:

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

Хоча директорія `$LFS/tools` може бути видаленою, коли система LFS буде закінченою, її можна використати для побудови додаткових систем LFS за цією версією книги. Як найкраще зробити резервне копіювання є справою персонально смаку.

> Застереження
>
> Якщо ви вирішили тримати тимчасові інструменти, щоб використовувати їх для побудови майбутніх систем LFS, зараз той час, щоб скопіювати їх як резерв на інший носій. Наступні команди у

Параграфі 6 будуть змінювати їх, надаючи їм неспроможність для майбутніх побудов.

Частина III. Будування системи LFS

Глава 6 Встановлення базового системного програмного забезпечення

6.1. Вступ

У цій частині, ми вступаємо у частину побудови і починаємо збирати систему LFS. Це означає, що ми використовуємо команду chroot для того, щоб використовувати тимчасову міні систему Лінукс, зробимо деякі фінальні підготовляння, і тоді розпочнемо встановлення пакетів.

Встановлення цього програмного забезпечення є простим. Хоча, у багатьох випадках, інструкції встановлення можуть бути коротшими і більш звичайними, ми вибрали повні інструкції для встановлення кожного пакету, щоб зменшити можливості помилок. Ключом до того, щоб вивчити, що змушує систему Лінукс працювати, є знати для чого кожен пакет використовується і чому ви (чи система) можуть його потребувати.

Ми не рекомендуємо використовувати оптимізацію. Вона може викликати пришвидшення програм, але вона може також спричинити утруднення компіляції і проблеми при виконанні програми. Якщо пакет відмовляється компілювати при використанні оптимізаційних опцій, спробуйте компілювати її без неї і перегляньте чи це виправило помилку. Навіть якщо пакет компілюється при використанні оптимізації, існує ризик, що вона скомпільована неправильно через комплекс взаємодій між кодом і інструментами побудови. Запам'ятайте також, що параметри `--march` і `-mtune` використовують значення, які не вказані у книзі, і не були тестовані. Це може спричинити проблеми з пакетами інструментів (Binutils, GCC і Glibc). Мала потенційна вигода від використання оптимізації компілятора часто переважується ризиком. Люди, які першими будували LFS, вказують на те, щоб будувати LFS без користувацької оптимізації. Вихідна система буде працювати дуже швидко і буде одночасно стабільною.

Порядок встановлення пакетів у даній частині книги, повинен категорично виконуватися, щоб запевнити, що ніяка програма випадково не набуде жорстко встановленого шляху, який вказуватиме на директорію `/tools`. Для цієї ж причини, не компілюйте окремі пакети паралельно. Паралельне компілювання може зберегти час (особливо на двоядерних машинах), однак це може спричинити жорстку прив'язання до шляху `/tools`, що призведе до зупинки роботи програми, коли ця директорія буде видаленою.

Перед інструкціями встановлення, кожна сторінка забезпечує інформацією про пакети, включаючи стислий опис того, що вони в собі вміщають, приблизний час побудови, і скільки дискового простору є необхідно для процесу встановлення. За інструкціями для встановлення, буде список програм і бібліотек, які встановлює пакет.

> Увага
>
> Значення SBU і необхідного дискового простору включає дані тестових наборів виконаних для усіх пакетів у Главі 6.

6.2. Підготовка віртуальної файлової системи ядра.

Деякі підтримувані ядром файлові системи використовуються для комунікації з ядром. Ці файлові системи є віртуальними у тому, що ніякого дискового простору вони не використовують. Вміст цих файлових систем розміщається у пам'яті.

Почніть з створення директорій у які дана файлова система буде змонтованою:

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

6.2.1. Створення внутрішніх файлів пристроїв

Коли ядро завантажує систему, воно вимагає присутність деяких файлів пристроїв. В особливості `console` і `null`. Файли пристроїв повинні бути створеними на жорсткому диску до того, як udevd буде запущеною, і, додатково, коли Лінукс запускається з параметром `init=/bin/bash`. Створіть файли пристроїв виконуючи наступні команди:

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

6.2.2. Монтування і заповнення /dev

Рекомендований метод заповнення директорії `/dev` пристроями, є монтування віртуальної файлової системи (такої як tmpfs) в директорію `/dev`, дозволити динамічно створювати пристрої у цій директорії як тільки вони виявлені чи доступні. Створення пристроїв зазвичай робиться під час процесу завантаження програмою Udev. Так як нова система ще не має Udev і ще не була завантаженою, необхідно монтувати і заповнити директорію `/dev` вручну. Це виконується монтуванням з прив'язкою директорії хостової системи. Монтування з прив'язкою є спеціальним типом монтування, яке дозволяє вам створювати віддзеркалення директорії чи точки монтування до іншого місця. Використовуйте наступну команду щоб досягнути цього:

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

6.2.3. Монтування віртуальних файлових систем ядра

Зараз змонтуйте згадані віртуальні файлові системи ядра:

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

У деяких хостових системах, /dev/shm є символічним посиланням до /run/shm. Всередині робочого середовища chroot, це символічне посилання повинно бути замінене на нормальну директорію, перед тим як монтувати тимчасову файлову систему:

```
if [ -h /dev/shm ]; then
     rm -f $LFS/dev/shm
     mkdir $LFS/dev/shm
fi
mount -vt tmpfs shm $LFS/dev/shm
```

6.3. Управління пакетами

Управління пакетами часто вимагається як додаток до книги LFS. Менеджер пакетів дозволяє відстежувати встановлення файлів, полегшуючи видалення і оновлення пакетів. Так як і з бінарними файлами і бібліотеками, менеджер пакетів обробить встановлення конфігураційних файлів. Перед тим як ви почнете дивуватись, ця секція не буде обговорювати і не рекомендувати який-небудь менеджер пакетів. Він забезпечує деяку кількість більш популярних технік. Ідеальний менеджер пакетів для вас може бути серед цих технік, або може бути комбінацією цих двох чи більше технік. Ця секція коротко згадує питання, які можуть виникнути під час оновлення пакетів.

Деякі причини, чому ніякий менеджер пакетів не був згаданий у LFS і BLFS включає наступні пункти:

- Робота з менеджером пакетів відносить фокус від цілі даної книги — вивчення побудови системи

Лінукс.

- Є багато рішень менеджерів пакетів, кожен має свої сильні сторони і слабкі. Вибрати тільки одного, який би задовольнив усю аудиторію, є складним.

Є деякі підказки написані в темі менеджера пакетів. Відвідайте Hints Project і подивіться чи один з них задовольняє ваші потреби.

6.3.1. Питання оновлення

Менеджер пакетів полегшує оновлення програм, коли виходять їхні новіші версії. В загальному інструкції у книгах LFS і BLFS можуть бути використані для того, щоб оновити програми до нових версій. Ось деякі пункти, які ви повинні знати коли оновлюєте пакети, особливо на запущеній системі.

- Якщо один з пакетів інструментарію (Glibc, GCC чи Binutils) повинен бути оновленим до новішої версії, безпечніше перебудувати LFS. Ви *можете* перебудовувати усі пакети у їхньому порядку залежностей, але ми цього не рекомендуємо. Для прикладу, якщо glibc-2.2.x повинна бути оновлена до glibc-2.3.x, буде безпечнішим перебудувати усе. Для оновлень мікроверсій, зазвичай працює просте перевстановлення, але не гарантовано. Для прикладу, оновлення від glibc-2.3.4 до glibc-2.3.5 зазвичай не спричинить ніяких проблем.

- Якщо пакети вміщують оновлення спільні бібліотеки і змінилось ім'я бібліотек, в такому випадку усі пакети, динамічно зв'язані з даними бібліотеками, повинні бути скомпільованими ще раз, для прив'язання до нових бібліотек. (Майте на увазі, що немає ніякого зв'язку між версіями пакетів і назвами бібліотек.) Для прикладу, дано пакет `foo-1.2.3`, що встановлює динамічну бібліотеку з назвою `libfoo.so.1`. Скажемо, що ви оновили пакет до нової версії `foo-1.2.4`, яка встановлює динамічну бібліотеку з назвою `libfoo.so.2`. У цьому випадку, усі пакети, що динамічно зв'язана з libfoo.so.1 повинні перекомпілюватись для зв'язування з `libfoo.so.2`. Зверніть увагу, що ви не повинні видаляти попередню бібліотеку поки динамічні бібліотеки не будуть скомпільованими ще раз.

6.3.2. Методи пакетних менеджерів

Наступне буде деякими поширеними техніками, які симулюють поведінку пакетних менеджерів. Перед тим як зупинитись на одному пакетному менеджері, зробіть розвідку по усіх можливих техніках, особливо по недолікам основних схем.

6.3.2.1. Усе знаходиться у моїй голові!

Так, це є технікою пакетного менеджера. Деякі люди не мають потреби у пакетному менеджері, адже вони тісно знайомі з усіма пакетами і які файли встановлюються кожним пакетом. Деякі користувачі також не потребують пакетних менеджерів через те, що вони планують перебудову усієї системи, коли змінюється пакет.

6.3.2.2. Встановлення у окремих директоріях

Це є простим методом контролю пакетів, який не вимагає ніяких унікальних пакетів (програм) для контролю встановлення. Кожен пакет встановлюється у окремій директорії. Для прикладу, пакет foo-1.1 встановлюється у `/usr/pkg/foo-1.1` і створюється посилання `/usr/pkg/foo` на `/usr/pkg/foo-1.1`. Коли встановлюється нова версія пакету foo — `foo-1.2`, вона встановлюється в директорію `/usr/pkg/foo-1.2` і попереднє посилання замінюється посиланням на нову версію.

Змінні середовища, такі як PATH, LD_LIBRARY_PATH, MANPATH, INFORMPATH і CPPFLAGS потрібно розширити, включенням шляху /usr/pkg/foo. Для більш ніж одного пакету, ця схема стає некерованою.

6.3.2.3. Управління пакетами шляхом посилань

Це є варіація попередньої техніки управління пакетами. Кожний пакет встановлюється аналогічно до попередньої схеми. Але замість того, щоб робити посилання на каталоги, кожен файл посилається в ієрархію каталогу /usr. Це забирає необхідність у зміні змінних середовища. Хоча посилання може зробити користувач для автоматизації, багато пакетних менеджерів були написаними використовуючи цей підхід. Найпопулярніші з них є Stow, Epkg, Graft і Depot.

Встановлення повинно бути фальшивим, щоб пакет думав, що він знаходиться у /usr, однак в реальності він встановлений у ієрархії /usr/pkg. Встановлення за даною манерою зазвичай не є тривіальним завданням. Для прикладу, припустимо, що встановлюєте пакет libfoo-1.1. Наступні інструкції не встановлять пакет правильно:

```
./configure --prefix=/usr/pkg/libfoo/1.1
make
make install
```

Встановлення буде працювати, але залежні пакети можуть не зможуть з'єднуватися як ви очікували. Якщо ви скомпілювали пакет який використовує бібліотеку libfoo, ви можете зазначити, що він використовує /usr/pkg/libfoo/1.1/lib.libfoo.so.1 замість /usr/lib/libfoo.so.1 як можна було б очікувати. Правильним підходом буде використання стратегії DESTDIR для фальшивої інсталяції пакету. Цей підхід працює наступним чином:

```
./configure --prefix=/usr
make
make DESTDIR=/usr/pkg/libfoo/1.1 install
```

Більшість пакетів підтримують цей підхід, але є і такі що не підтримують. Для пакетів які не відповідають загальній структурі, ви можете або вам буде необхідно встановлювати пакет вручну, або краще встановити деякі тяжкі пакети у директорію /opt.

6.3.2.4. Основані на часових мітках (timestamp)

За цією технікою, на файл накладається мітка часу перед встановленням пакету. Після встановлення, просте використання команди **find** з необхідними опціями може генерувати журнал усіх файлів встановлених після того як був встановлений файл з відповідною часовою міткою. Менеджер пакетів написаний за таким підходом є встановлювачами з журналами.

Хоча ця схема має перевагу у простоті, вона має два недоліки. Якщо під час встановлення, файли встановлюються з іншою часовою міткою ніж теперішній час, ці файли не будуть враховані інсталятором. Також, ця схема може бути використаною коли один пакет встановлюється за одиницю часу. Журнали не є надійними, якщо два пакети були встановленими з двох різних консолей.

6.3.2.5. Простежування за скриптами встановлення

У цьому підході, команди, які виконує скрипт встановлення, записуються. Є дві техніки, з яких одну можна використати:

Встановлюється змінна середовища LD_PRELOAD, яка вказує на бібліотеку, яка повинна бути попередньо

завантаженою перед встановленням. Під час встановлення, ця бібліотека записує пакети які були встановленими, прикріплюючи себе до виконуваних файлів, таких як **cp, install, mv** і прослухуючи системні виклики які змінюють файлову систему. Для того щоб цей метод працював, усі виконувані файли повинні бути динамічно зв'язаними без бітів suid і sgit. Перед-завантажування бібліотеки, може викликати деякі небажані побічні ефекти під час встановлення. Однак, рекомендується виконати перевірку для запевнення, що менеджер пакетів не ламає що-небудь і записує в журнал усі відповідні файли.

Друга техніка полягає у використанні **strace**, що записує в журнал усі системні виклики під час виконання скриптів встановлення.

6.3.2.6. Створення пакетних архівів

У цій схемі, встановлення пакетів відбувається у окремі дерева директорій, як було описано у Управлінні пакетами шляхом посилань. Після встановлення, створюються архіви пакетів з встановлених файлів. Цей архів використовується для встановлення пакетів на локальній машині або може бути використано для встановлення пакету на інших машинах.

Цей підхід використовується більшістю менеджерами пакетів комерційних дистрибутивів. Приклад менеджера пакетів, який користується даним методом є RPM (що, випадково, вимагається стандартом Linux Standart Base Specification), pkg-utils, Debian apt і система портів Gentoo. Підказку, як пристосувати ці стилі пакетних менеджерів для систем LFS розміщується за адресою `http://www.linuxfromscratch.org/hints/downloads/files/fakeroot.txt`.

Створення пакетних файлів, які вміщають інформацію про залежності є складним у з рамками LFS.

Дистрибутив Slackware використовує основані **tar** системи пакетних архівів. Ця система не вирішує пакетні залежності, як це роблять більш комплексні менеджери. Для більш детальної інформації про пакетний менеджер Slackware, перегляньте сторінку за адресою `http://www.slackbook.org/html/package-management.html`.

6.3.2.7. Основані користувачами менеджмент

Ця схема, унікальна для LFS, була створена Маттіас Бенкман (Matthias Benkmann), і є доступною з Hints Project. У цій схемі, кожен пакет встановлюється як окремий користувач в стандартні місця. Файли, які належать до пакету, легко ідентифікувати перевіряючи ID користувача. Можливості і недоліки цього методу є надто комплексні для опису їх у цій секції. Для більшої інформації перейдіть до підказки за адресою `http://www.linuxfromscratch.org/hints/downloads/files/more_control_and_pkg_man.txt`.

6.3.3. Розгортання LFS на багатьох системах.

Одна з переваг систем LFS у тому, що немає файлів, які б залежали від позиції інших файлів на диску. Копіюючи побудову LFS на інший комп'ютер, з подібною архітектурою до базової системи є так само просто, як і використання tar на диску LFS, що вміщує кореневу директорію (приблизно 250 Мбайт архівованих файлів базової побудови LFS), копіюючи цей файл через мережу чи CD-ROM до нової системи. З цієї точки, деякі конфігураційні файли мають бути змінені. Конфігураційні файли, що можуть потребувати оновлення: `/etc/hosts`, `/etc/fstab`, `/etc/passwd`, `/etc/group`, `/etc/shadow`, `/etc/ld.so.conf`, `/etc/sysconfig/rc.site`, `/etc/sysconfig/network`, і `/etc/sysconfig/ifconfig.eth0`.

Ядро користувача може потребувати перебудови, в залежності від конфігурації апаратного забезпечення і

оригінальної конфігурації ядра.

На кінець, нова система повинна бути здатною до завантаження використовуючи Секцію 8.4, “Використання GRUB для встановлення процесу завантаження”.

6.4. Використовуємо нове робоче середовище через команду chroot

Настав час увійти в середовище через команду chroot, щоб розпочати побудову і встановлення фінальної системи LFS. Як користувач root, виконайте наступну команду для входження в область яка є, на даний момент, заповнена тимчасовим інструментарієм:

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

Опція -i, дана до команди env очистить усі змінні середовища chroot. Після цього, тільки змінні `HOME`, `TERM`, `PS1` і `PATH` будуть встановленими. Конструкція `TERM=$TERM` встановить змінну TERM всередині середовища chroot у таке саме значення як і поза ним. Ця змінна необхідна для програм, таким як **vim** і **less,** для правильної роботи. Якщо інші змінні необхідні, такі як `CFLAGS` чи `CXXFLAGS`, це є хороше місце, щоб встановити їх знову.

З цієї точка і надалі, немає більше необхідності використовувати змінну LFS тому, що уся робота буде відбуватися на файловій системі LFS. Це відбувається тому, що оболонка Bash “знає” нову кореневу директорію (/).

Запам'ятайте, що `/tools/bin` проходить останньою у змінній `PATH`. Це означає, що тимчасові інструменти не будуть більше використовуватися, якщо фінальна версія вже встановлена. Це відбувається коли командна оболонка не “запам'ятовує” місце знаходження виконуваних файлів — для цієї причини, хешування вимкнено параметром `+h` у **bash**.

Запам'ятайте, що оболонка **bash** скаже вам “I have no name!”. Це є нормально, так як файл `/etc/passwd` ще не був створений.

> Увага
>
> Дуже важливо виконувати усі команди, в цьому параграфі і наступних, у середовищі chroot. Якщо ви покинете це середовище з будь-яких причин (перевантаження, наприклад), впевніться, що віртуальна файлова система є змонтована, як описано у Секції 6.2.2 “Монтування і заповнення /dev”, Секції 6.2.3, “Монтування віртуальної файлової системи ядра” і увійдіть за в новостворене середовище за допомогою команди **chroot**, для продовження встановлення.

6.5. Створення директорій

Настав час створити деяку структуру у системі LFS. Створіть стандартне дерево каталогів виконуючи наступні команди:

```
mkdir -pv /{bin,boot,etc/{opt,sysconfig},home,lib,mnt,opt,run}
mkdir -pv /{media/{floppy,cdrom},sbin,srv,var}
install -dv -m 0750 /root
install -dv -m 1777 /tmp /var/tmp
```

```
mkdir -pv /usr/{,local/}{bin,include,lib,sbin,src}
mkdir -pv /usr/{,local/}share/{doc,info,locale,man}
mkdir -v /usr/{,local/}share/{misc,terminfo,zoneinfo}
mkdir -pv /usr/{,local/}share/man/man{1..8}
for dir in /usr /usr/local; do
ln -sv share/{man,doc,info} $dir
done
case $(uname -m) in
x86_64) ln -sv lib /lib64 && ln -sv lib /usr/lib64 ;;
esac
mkdir -v /var/{log,mail,spool}
ln -sv /run /var/run
ln -sv /run/lock /var/lock
mkdir -pv /var/{opt,cache,lib/{misc,locate},local}
```

Директорії є, за замовчуванням, створювані з правами доступу 755, але це не є добрим для усіх директорій. У вищенаведених командах, зроблено дві зміни: одна для домашньої директорії користувача root, і інша для директорій тимчасових файлів.

Перша зміна прав запевняє, що не просто усі можуть зайти в директорію `/root` — так само, як і зі своїми директоріями звичайні користувачі. Друга зміна прав забезпечує можливість запису для кожного користувача у каталозі `/tmp` і `/var/tmp`, але вони не можуть видалити файли користувачів з них. Остання робиться за допомогою так званого “липучого біту” (“sticky bit”), найвищий біт у бітовій масці 1777.

6.5.1. Підтримка FHS

Дерево каталогів базується на стандарті FHS (Filesystem Hierarchy Standart — стандарт файлової системи) (доступний за адресою *http://www.pathname.com/fhs/*). В добавок до FHS, ми створили сумісні посилання для каталогів man, doc і info так як багато пакетів досі намагаються встановити їхню документацію в `/usr/<directory>` чи `/usr/local/<directory>` замість `/usr/share/<directory>` і `/usr/share/games`. FHS не специфікує структуру підкаталогу `/usr/local/share`, отож ми створюємо тільки каталоги, які є необхідними. Однак, у вас є свобода створювати ці директорії якщо ви надаєте перевагу відповідати стандарту FHS більш точно.

6.6. Створення істотних файлів і посилань

Деякі програми використовують жорстко-вмонтовані шляхи до програм, які ще не існують у нашій системі. Для того щоб задовольнити ці програми, створіть декілька символічних посилань, які будуть замінені на справжні програми, по виконанні курсу цієї глави після того як усе програмне забезпечення буде встановлено:

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

Належні Лінукс системи підтримують список монтованих файлових систем у файлі `/etc/mtab`. Нормально, цей файл буде створений, коли ми змонтуємо нові файлові системи. Так як ми не будемо монтувати які-небудь розділи всередині нашого середовища chroot, створіть пустий файл для утиліт, які очікують присутності файлу `/etc/mtab`:

```
touch /etc/mtab
```

Для того, щоб користувач root мав змогу заходити в систему і ім'я “root” розпізнавалося, має бути відповідні записи у файлах /etc/passwd і /etc/group.

Створіть файл /etc/passwd виконуючи наступну команду:

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

Самий пароль для root (“x” використовується просто для займання місця) буде встановлено пізніше.

Створіть файл /etc/group виконуючи наступну команду:

```
cat > /etc/group << "EOF"
root:x:0:
bin:x:1:
sys:x:2:
kmem:x:3:
tape:x:4:
tty:x:5:
daemon:x:6:
floppy:x:7:
disk:x:8:
lp:x:9:
dialout:x:10:
audio:x:11:
video:x:12:
utmp:x:13:
usb:x:14:
cdrom:x:15:
mail:x:34:
nogroup:x:99:
EOF
```

Створені групи не є частиною якого-небудь стандарту — ці групи створюються через те, що вони є необхідними для конфігурування Udev у цьому розділі, і також є звичаєм впровадженим числом існуючих дистрибутивів Лінукс. Стандарт LSB рекомендує тільки це, крім присутності групи root з ID групи (GID) встановленим у 0 (нуль), групи bin з GID встановленим у 1. Усі інші імена груп і їхніх GID може бути вільно вираним системним адміністратором, так як добре написані програми не залежать від номерів GID, але швидше за все використовує ім'я групи.

Для того щоб видалити рядок “I have no name!”, розпочніть нову сесію оболонки. Так як повна Glibc була встановлена в Частині 5 і файли /etc/passwd і /etc/group були створені, механізм імен користувачів і груп тепер буде працювати:

```
exe /tools/bin/bash —login +h
```

Запам'ятайте використовувати директиву +h. Вона вказує **bash** не використовувати внутрішній механізм хешування шляхів. Без цієї директиви, **bash** запам'ятає шляхи до бінарних файлів, які він виконав. Щоб впевнитися в використанні нових скомпільованих програм одразу після їх встановлення, директива +h буде використаною під час цієї частини.

Програми **login**, **agetty** і **init** (і інші) використовують деякі файли журналів для запису інформації, такої як хто заходив в систему і коли. Однак, ці програми не будуть робити записи у журнали, якщо вони не існують.

Ініціалізуйте файли журналів і дайте їм відповідні права:

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

У файл /var/log/wtmp записується інформація про усі входи і виходи з системи. У файл /var/log/lastlog записується інформація часу останнього входу в систему кожного користувача. У файл /vat/log/btmp записується інформація невдалих спроб входу.

> Увага
>
> Файл /run/utmp зберігає інформацію про користувачів, які знаходяться у системі. Ці файли створені динамічно скриптами завантаження системи.

6.7. Заголовкові файли API linux-3.5.2

Заголовкові файли API Лінукса постачають інтерфейс до ядра, який використовується Glibc.

Приблизний час побудови: 0.1 SBU
Необхідний дисковий простір: 515 Мбайт

6.7.1. Встановлення заголовкових файлів Лінукс

Ядро Лінукс має надавати інтерфейс програмування додатків (API) для системної бібліотеки мови C (Glibc у LFS). Це робиться шляхом витягування деяких заголовкових файлів мови C, які розміщені у архіві коду ядра Лінукса.

Впевніться, що немає ніяких сторонніх файлів і залежностей у дереві коду з попередньої активності:

```
make mrproper
```

Зараз, протестуйте і витягніть доступні для користувача заголовкові файли ядра з дерева коду. Вони розміщені у проміжному локальному каталозі і будуть скопійованими в необхідне місце тому, що процес витягування видаляє усі існуючі файли у цільовому каталозі. Завжди є деякі приховані файли, які використовуються розробниками ядра і не потрібні для LFS, що видаляються з проміжного каталогу.

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

6.7.2. Вміст заголовкових файлів Лінукс

Встановлені заголовкові файли:	/usr/include/asm/*.h, /usr/include/asm-generic/*.h, /usr/include/drm/*.h, /usr/include/linux/*.h, /usr/include/mtd/*.h, /usr/include/rdma/*.h, /usr/include/scsi/*.h, /usr/include/sound/*.h, /usr/include/video/*.h, /usr/include/xen/*.h
Встановлені каталоги:	/usr/include/asm, /usr/include/asm-generic, /usr/include/drm, /usr/include/linux, /usr/include/mtd, /usr/include/rdma, /usr/include/scsi, /usr/include/sound, /usr/include/video, /usr/include/xen

Короткий опис

/usr/include/asm/*.h	Заголовкові файли API Лінукс ASM
/usr/include/asm-generic/*.h	Загальні заголовкофі файли Лінукса API ASM
/usr/include/drm/*.h	Заголовкові файли Лінукс API для DRM
/usr/include/linux/*.h	Заголовкові файли Лінукс API
/usr/include/mtd/*.h	Заголовкові файли Лінукс API для MTD

`/usr/include/rdma/*.h`	Заголовкові файли Лінукс API для RDMA
`/usr/include/scsi/*.h`	Заголовкові файли Лінукс API для SCSI
`/usr/include/sound/*.h`	Заголовкові файли Лінукс API для звуку
`/usr/include/video/*.h`	Заголовкові файли Лінукс API для відео
`/usr/include/xen/*.h`	Заголовкові файли Лінукс API для Xen

6.8. Man-pages-3.42

Пакет Man-pages вміщує більш ніж 1900 сторінок допомоги.

Приблизний час побудови: менше ніж 0.1 SBU
Необхідний дисковий простір: 22 Мбайт

6.8.1. Встановлення Man-pages

Встановіть пакет, виконуючи:

```
make install
```

6.8.2. Вміст Man-pages

Встановлені файли: певна кількість сторінок допомоги

Короткий опис

`сторінки man` Описують мову програмування C, важливі файли пристроїв і значні конфігураційні файли.

6.9. Glibc-2.16.0

Пакет Glibc вміщає головну бібліотеку мови C. Ця бібліотека забезпечує базові функції для виділення пам'яті, пошуку каталогів, відкриття і закриття файлів, читання і запис файлів, обробку рядків, пошук за шаблоном, арифметичні функції і так далі.

Приблизний час побудови: 17.6 SBU
Необхідний дисковий простір: 852 Мбайт

6.9.1. Встановлення Glibc

> Увага
>
> Деякі пакети ззовні LFS пропонують встановити GNU libiconv для того щоб перетворити дані з одного кодування в інше. Домашня сторінка проекту (http://www.gnu.org/software/libiconv/) каже: “Ця бібліотека постачає реалізацію функції iconv(), для використання на системах, які не мають таку, або на системах, які не можуть перетворити з/у Unicode”. Бібліотека Glibc постачає htfkspfws. функції iconv(), яка може перетворювати з/у Unicode, отож libiconv не є потрібною на системах LFS.

Системи побудови Glibc є самодостатніми і встановлюються ідеально, навіть якщо функціональні файли компілятора і компонувальника як і раніше розміщуються у /tools. Функціональні файли і компонувальник не можуть бути сконфігурованими перед встановленням Glibc тому, що тест autoconf бібліотеки Glibc дасть неправильні результати і не дозволить отримати чисту побудову.

Виправте проблему, яка спричиняє провал побудови у середовищі LFS:

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

Коли ви виконуєте make install, скрипт, який називається test-installation.pl виконує малий розумний тест нашої ново-встановленої Glibc. Однак, він має помилку, яка спричиняє провал цих тестів, отож, для того, щоб запобігти його виконання виконайте наступну команду:

```
sed -i '/test-installation.pl/d' Makefile
```

Скрипт командної оболонки ldd вміщує специфічний для Bash синтакс. Встановіть його програму-інтерпретатор за замовчуванням /bin/bash у випадку іншого встановленного за посиланням /bin/sh, як було описано у частині про командні оболонки книги BLFS:

```
sed -i 's|@BASH@|/bin/bash|' elf/ldd.bash.in
```

Зараз виправте проблему, яка спричиняє деякі крахи програм при використанні проблематичних доменних імен.

```
patch -Np1 -i ../glibc-2.16.0-res_query_fix-1.patch
```

Документація Glibc рекомендує будування Glibc ззовні кореневого каталогу коду у спеціально-створені директорії.

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

Підготуйте Glibc до встановленння:

```
../glibc-2.16.0/configure \
    --prefix=/usr \
    --disable-profile \
    --enable-add-ons \
```

```
    --enable-kernel=2.6.25 \
    --libexecdir=/usr/lib/glibc
```

Значення нових опцій конфігурування:

--libexecdir=/usr/lib/glibc

Це змінює розміщення програми pt_chown з звичайної /usr/libexec у /usr/lib/glibc.

Скомпілюйте пакет

```
make
```

> Важливо
>
> У цій секції, виконання пакету тестів є критично важливим. Не пропускайте його виконання ні під якими обставинами.

В загальному, деякі тести не проходять, але ви можете проігнорувати деякі їхні помилки, які перераховані нижче. Зараз протестуйте результати побудови:

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

Ви ймовірно побачите очікувані (для ігнорування) помилки тестів /posix/annexc і conform/run-conformtest. Набір тестів Glibc в деякій мірі залежний від хостової системи. Ось список найбільш поширених проблем:

- Відомо, що тести *nptl/tst-clock2, nptl/tst-attr3, tst/tst-cputimer1* і *rt/tst-cpuclock2* проваляться. Причина цього не достатньо зрозуміла, але є свідчення, що незначні проблеми можуть спричинити ці проблеми.
- Математичний тест інколи падає поки виконується на системі в якій процесор не є відносно новим справжнім Intel чи автентичним AMD процесором.
- Коли тест виконується на системі з старішим і повільнішим апаратним забезпеченням, яка є завантаженою, може провалитися тому, що виходить час очікування. Модифікуючи команду make check встановленням TIMEOUTFACTOR, допомагає ліквідувати ці помилки (**TIMEOUTFACTOR=16 make -k check**).
- Інші тести які провалюються на деяких архітектурах є posix/bug-regex32, misc/tst-writev, elf/check-textrel, nptl/tst-getpid2 і stdio-common/bug22.

Хоча це не шкідливе повідомлення, стадія встановлення Glibc буде скаржитись на відсутність `/etc/ld.so.conf`.

Виключіть це попередження за допомогою:

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

Встановіть пакет

```
make install
```

Встановіть пов'язані заголовкові файли NIS і RPC, які не встановлюються за замовчуванням; ці файли необхідні для перебудови glibc і деяким пакетам BLFS:

```
cp -v ../glibc-2.16.0/sunrpc/rpc/*.h /usr/include/rpc
cp -v ../glibc-2.16.0/sunrpc/rpcsvc/*.h /usr/include/rpcsvc
```

```
cp -v ../glibc-2.16.0/nis/rpcsvc/*.h /usr/include/rpcsvc
```

Локалі, які можуть дати можливість системі відповідати у різних мовах, не встановлені цією командою. Ніякі з локалей не є потрібними, але, якщо деякі з них відсутні, тестові набори майбутніх пакетів пропустять важливі тести.

Індивідуальні локалі можуть бути встановленими, використовуючи програму localdef. Тобто, перша команда localdef нижче, комбінує незалежне локальне кодування /usr/share/i18n/locales/cs_CZ з таблицею символів /usr/share/i18n/charmaps/UTF-8.gz і викладає результат у файл /usr/lib/locale/locale-archive . Наступні інструкції встановлять мінімальний набір локаль необхідних для оптимального покриття тестів:

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

В додаток, встановіть локаль для вашої країни, мови і набору символів.

Альтернативно, ви можете встановити усі локалі, які перераховані у файлі glibc-2.16.0/localedata/SUPPORTED (він включає усі перераховані вище локалі і набагато більше) однією трудомісткою командою:

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

Після цього використовуйте команду localdef для створення і встановлення локалей, які не перераховані у файлі glibc-2.16. 0/localedata/SUPPORTED , крім випадків їх непотрібності вам.

6.9.2. Конфігурація Glibc

Файл /etc/nsswitch.conf повинен бути створеним тому, що, хоча Glibc постачає стандарти, коли цей файл відсутній або пошкоджений, стандарти Glibc не працюють добре у мережному середовищі. Часова зона також повинна бути сконфігурованою.

Створіть новий файл /etc/nsswithc.conf виконуючи наступне:

```
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
```

Встановіть дані часового поясу:

```
tar -xf ../tzdata2012e.tar.gz
ZONEINFO=/usr/share/zoneinfo
mkdir -pv $ZONEINFO/{posix,right}
for tz in etcetera southamerica northamerica europe africa antarctica \
asia australasia backward pacificnew solar87 solar88 solar89 \
systemv; do
zic -L /dev/null
-d $ZONEINFO
-y "sh yearistype.sh" ${tz}
zic -L /dev/null
-d $ZONEINFO/posix -y "sh yearistype.sh" ${tz}
zic -L leapseconds -d $ZONEINFO/right -y "sh yearistype.sh" ${tz}
done
cp -v zone.tab iso3166.tab $ZONEINFO
zic -d $ZONEINFO -p America/New_York
unset ZONEINFO
```

Значення команд zic:

zic -L /dev/null ...

Ця створює часові зони posix, без будь-яких стрибучих секунд. Зазвичай вони вкладаються у обидва каталоги zoneinfo і zoneinfo/posix. Необхідно викладати часові пояси POSIX у директорію zoneinfo, інакше деякі набори тестів будуть видавати помилки. У вбудованих систем, де простір не є великим і ви не маєте намір оновлювати часові пояси, ви можете зберегти 1.9 Мбайт, не використовуючи директорію posix, але деякі програми чи тестові набори можуть видавати погані результати.

zic -L leapseconds ...

Ця команда створює правильні часові пояси, включаючи стрибучі секунди. У вбудованих систем, де є обмежений простір і ви не будете оновлювати часові зони чи дбати про коректний час, ви можете зберегти 1.9 Мбайт пропускаючи каталог right.

zic ... -p ...

Ця команда створює файл posixrules. Ми використовуємо New York тому, що стандарт POSIX вимагає відповідність правил збереження частин дня з првилами US.

Єдиним шляхом, щоб визначити локальний часовий пояс — виконати наступну команду:

```
tzselect
```

Після того, як ви відповісте на деякі запитання про своє місцезнаходження, срипт видрукує назву часового поясу (наприклад, Europe/Kyiv). Також існуюте декілька інших можливих часових поясів, які перераховані у каталозі /usr/share/zoneinfo, такі як Canada/Eastern чи EST5EDT, котрі не ідентифікуються скриптом, але

можуть бути використаними.

Після цього створіть файл /etc/localtime, виконуючи команду:

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

Замініть <xxx> ім'ям вибраного вами часового поясу.

Значення опцій команди cp:

--remove-destination

Ця опція є необхідною для примусового видалення вже існуючого символічного посилання. Причина копіювання файлу, замість того, щоб використовувати посилання в покриванні ситуації, коли дерево каталогів /usr розміщується на іншому розділі. Це може бути важливо коли система завантажується в режим одного користувача.

6.9.3. Конфігурація динамічного завантажувача.

Зазвичай, динамічний завантажувач (/lib/ld-linux.so.2) шукає у /lib і /usr/lib динамічні бібліотеки, які є необхідними програмами при виконанні. Однак. Якщо є бібліотеки у інших каталогах ніж /lib і /usr/lib, їх необхідно додати до файлу /etc/ld.so.conf для того, щоб динамічний завантажувач знайшов їх. Дві директорії, які ззазвичай містять додаткові бібліотеки є /usr/local/lib і /opt/lib, отож додайте ці каталоги до шляхів пошуку динамічного завантажувача.

Створіть новий файл /etc/ld.so.conf, виконуючи:

```
cat > /etc/ld.so.conf << "EOF"
# Begin /etc/ld.so.conf
/usr/local/lib
/opt/lib

EOF
```

За бажанням, динамічний завантажувач може також шукати директорії і включати контент файлів знайдених там. Зазвичай, файли в цій директорії являють собою рядок, який визначає шлях пошуку директорій. Для того. Щоб додати цю можливість, виконайте наступну команду:

```
cat >> /etc/ld.so.conf << "EOF"
# Add an include directory
include /etc/ld.so.conf.d/*.conf

EOF
mkdir /etc/ld.so.conf.d
```

6.9.4. Вміст Glibc

Встановлені програми: catchsegv, gencat, getconf, getent, iconv, iconvconfig, ldconfig, ldd, lddlibc4, locale, localedef, mtrace, nscd, pcprofiledump, pt_chown, rpcgen, sln, sotruss, sprof, tzselect, xtrace, zdump і zic

Встановлені бібліотеки: ld.so, libBrokenLocale.{a,so}, libSegFault.so, libanl.{a,so}, libbsd-compat.a, libc.{a,so}, libc_nonshared.a, libcidn.so, libcrypt.{a,so}, libdl.{a,so}, libg.a, libieee.a, libm.{a,so}, libmcheck.a, libmemusage.so, libnsl.{a,so},

libnss_compat.so, libnss_dns.so, libnss_files.so, libnss_hesiod.so, libnss_nis.so, libnss_nisplus.so, libpcprofile.so, libpthread.{a,so}, libpthread_nonshared.a, libresolv.{a,so}, librpcsvc.a, librt.{a,so}, libthread_db.so, and libutil.{a,so}

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

Короткий опис

catchsegv Може бути використоною для створення образу стеку, коли програма закінчується з помилкою segmentation fault

getcat Генерує каталоги повідомлень

getconf Виводить спечифічні для файлової системи значення системниї конфігурації

getent Дістає входження з адміністративної бази даних

iconv Робить перетворення між наборами символів

iconvconfig Створє швидкозавантажувані модульні файли конфігурації **iconv**

ldconfig Конфігурує зв'язки динамічного завантажувача під час виконання

ldd Повідомляє, які динамічні бібліотеки необхідні для данї програми чи динамічної бібліотеки

lddlibc4 Допомагає **ldd** з роботою над об'єктними файлами

locale Виводить деяку інформацію про поточну локаль

localedef Компілює локальні специфікації

mtrace Читає і інтерпретує файл образу пам'яті і виводить результат у форматі прийнятному для людей.

nscd Демон, який постачає кеш для найбільш поширених запитів сервісних імен

pcprofiledump утиліта, яка робить дамп інформації, згенерованої профілюванням PC

pt_chown Допоміжна програма для **grantpt** для встановлення власнка, групи і права доступу для похідного псевдо-терміналу.

rpcgen	Генерує код мови C для реалізації протоколу Remote Procedure Call (RPC)
sln	Статично компонована програма **ln**
sotruss	Трасує виклики процедур динамічної бібліотеки даної команди
sprof	Читає і виводить дані профілю динамічної бібліотеки
tzselect	Питає користувача про розміщення системи і виводить відповідний опис часового поясу
xtrace	Трасує виконання програми, друкуючи поточну виконану функцію
zdump	Дампер файлів часового поясу
zic	Компілятор файлів часового поясу
`ld.so`	Допоміжна програма для виконуваних файлів динамічних бібліотек
`libBrokenLocale`	Для внутрішніх потреб Glibc, як засіб для визначення поламаних виконуваних програм. Дивіться коментарії у файлі glibc-2.16.0/locale/broken_cur_ max.c для детальнішої інформації
`libSegFault`	Обробник сигналу segmentation fault, використовується **catchsegv**
`libanl`	Асинхронна бібліотека пошуку імен
`libbsd-compat`	Постачає необхідну сумісність для виконання деяких програм Berkeley Software Distribution (BSD) під Лінукс.
`libc`	Головна бібліотека мови C
`libcidn`	Бібліотека для внутрішнього використання Glibc, виконує обробку інтернаціональних доменних імен у функції getaddrinfo()
`libcrypt`	Бібліотека криптографії
`libdl`	Бібліотека інтерфейсу динамічного компонувальника
`libg`	Пуста бібліотека без яких-небудь функцій. Попередньо використовувалась як виконувана бібліотека для **g++**
`libieee`	Компонування у цьому модулі змушує правила обробки помилок як визначено у Institute of Electrical and Electronic Engineers (IEEE). За замовчуванням це обробка помилок POSIX.1
`libm`	Математична бібліотека
`libmcheck`	Вмикає перевірку виділення пам'яті, коли до неї компонуються

libmemusage	Використовується **memusage** для допомоги збирання інформації про використання пам'яті програмою
libnsl	Бібліотека мережних сервісів
libnss	Бібліотека Name Service Switch (вибір сервісного ім'я), яка вміщує функції для отримання імен хостів, імен користувачів, груп, псевдонімів, сервісів, протоколів і т.д.
libpcprofile	Вміщує функції профілювання, які використовуються для відстеження суми використання часу процесора у даних рядках коду
libpthread	Бібліотека потоків POSIX
libresolv	Вміщує функції для створення, посилання і інтерпретування пакетів до серверів доменних імен Інтернету.
librpcsvc	Вміщує функції, які забезпечують сервіси RPC
librt	Вміщує функції, які забезпечують більшість інтерфейсів специфікованих стандартом POSIX.1b Realtime Extension
libthread_db	Містить корисні функції для побудови відлагоджувачів для багатопотокових програм
libutil	Вміщує код для “стандартних” функцій, які використовуються у різних утилітах Юнікс (Unix)

6.10. Налаштування набору інструментів

Зараз, коли фінальні бібліотеки мови C встановлені, настав час налагодити інструментарій так, щоб він компонував будь-яку нову скомпільовану програму з цими новими бібліотеками.

Для початку, зробіть резервну копію компонувальника у `/tools`, і замініть його налаштованим компонувальником, який ми зробили у главі 5. Ми також створемо посилання до його дублікату у `/tools/$(gss -dumpmachine)/bin`:

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

Наступним, внесемо поправки у функціональні файли GCC так, щоб він використовував новий динамічний компонувальник. Простим видаленням усіх входжень “/tools” повинно дати нам правильний шлях до динамічного компонувальника. Також налаштуйте функціональні файли так, щоб GCC “знав” де знайти правильні заголовкові файли і початкові файли Glibc. Команда **sed** виконає це:

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

Це буде хорошою ідеєю візуально перевірити функціональні файли щоб встановити чи виконалися призначені зміни.

У цій точці важливо переконатися, що базові функції (компілювання і компонування) налаштованого інструментарію працює як очікується. Щоб це зробити, виконайте наступну перевірку:

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

Якщо усе працює коректно, не повинно бути ніяких помилок і вивід останньої команди буде виглядати (згідно з залежними від платформи відмінності імені динамічного компонувальника):

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

Зверніть увагу, що /lib є префіксом нашого динамічного компонувальника.

Зараз впевніться, що ми встановили використання правильних початкових файлів:

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

Якщо все працює правильно, неповинно бути яких-небудь помилок, і вивід останньої команду буде виглядати як:

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

Перевірте чи компілятор шукає коректні заголовкові файли:

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

Команда повинна завершитись успішно з наступним виводом:

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

Наступним чином, перевірте чи компонувальник використовується з коректними шляхами пошуку:

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

Якщо усе працює правильно, не повинно бути ніяких помилок і вивід останньої команди (у відповідності до залежних від платформи цільових трійок) повинен бути:

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

Наступним, впевніться, що ми використовуємо коректну libc:

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

Якщо усе працює правильно, не повинно бути ніяких помилок і вивід останньої командт (у відповідності до каталогу lib64 на 64-бітних хостах) повинен бути:

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

І на кінець, впевніться, що GCC використовує правильний динамічний компонувальник:

```
grep found dummy.log
```

Якщо усе працює правильно, не повинно бути ніяких помилок і вивід останньої командт (у відповідності до платформенно залежних різниць між іменами компонувальника і каталогу lib64 на 64-бітних хостах) повинен бути:

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

Якщо вивід не з'явився як було показано вище або не надійшов взагалі, тоді щось критично неправильно. Розслідуйте і пройдіть ще раз усі кроки, щоб визначити проблему і виправте її. Найбільш ймовірною причиною неправильної поведінки з налаштовуванням функціональних файлів. Будь-яка проблема повинна бути вирішеною перед тим, як продовжувати процес.

Якщо усе працює коректно, очистіть тестові файли:

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

6.11. Zlib-1.2.7

Пакет Zlib вміщає підпрограми компресії і декомпресії, які використовуються деякими програмами.

Приблизний час побудови: 0.1 SBU
Необхідний дисковий простір: 3.9 Мбайт

1.2.7. Встановлення Zlib

Підготуйте Zlib до компіляції

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

Скомпілюйте пакет:

```
make
```

Для тестування результатів, виконайте:

```
make check
```

Встановіть пакет:

```
make install
```

Динамічна бібліотека повинна бути переміщена у /lib і файл з розширенням .so у /usr/lib/ повинен бути створений заново:

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

6.11.2. Вміст Zlib

Встановлені бібліотеки: lib.{a,so}

Короткий опис

libz	Містить функції компресії і декомпресії, які використовують деякі програми

6.12. File-5.11

Пакет File містить утиліту для визначення типу даного їй файлу чи файлів.

Приблизний час побудови: 0.1 SBU
Необхідний дисковий простір: 12.5 Мбайт

6.12.1. Встановлення File

Підготуйте File до компіляції

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

Скомпілюйте пакет:

```
make
```

Для тестування результатів, виконайте:

```
make check
```

Встановіть пакет:

```
make install
```

6.12.2 Вміст File

Встановлені програми: file

Встановлені бібліотеки: libmagic.{a,so}

Короткий опис

file	Намагається класифікувати кожен даний їй файл; вона робить це виконуючи декілька тестів — тести файлової системи, тести чарівних чисел і мовні тести
`libmagic`	Містить підпрограми для розпізнавання магічних чисел, використовується програмою **file**

6.13. Binutils-2.22

Пакет Binutils містить компонувальник, асемблерний компілятор, і інші інструменти для обробки об'єктних файлів.

Приблизний час побудови: 1.9 SBU
Необхідний дисковий простір: 343 Мбайт

6.13.1. Встановлення Binutils

Перевірте чи псевдо-термінали (PTYs) працюють всередині середовища chroot, виконуючи простий тест:

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

Ця команда повинна виводити наступне:

```
spawn ls
```

Якщо, замість того, вивід включає це повідомлення, тоді середовище не встановлене для правильних операцій з PTY. Це питання повинне бути вирішене перед тим як виконувати тестові набори для Binutils і GCC:

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

Забороніть встановлення застарілих файлів standards.info так як нові будуть встановлені пізніше у інструкціях Autoconf:

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

Застосуйте патч для запобігання помилок побудови, коли ви використовуєте оптимізації компілятора:

```
patch -Np1 -i ../binutils-2.22-build_fix-1.patch
```

Документація Binutils рекомендує будувати Binutils ззовні кореневого каталогу дерева коду у спеціально створеній директорії:

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

Підготуйте Binutils до компіляції

```
../binutils-2.22/configure --prefix=/usr --enable-shared
```

Скомпілюйте пакет:

```
make tooldir=/usr
```

Значення параметру make:

tooldir=/usr

За нормальних обставин, директорія інструментів (tooldir) встановлена у `$(exec_prefix)/$(target_alias)`. Для прикладу, машини x86_64 будуть розширювати її до `/usr/x86_ 64-unknown-linux-gnu`. Через те, що це є сконфігурована користувачем система, ця платформенно специфічна директорія у `/usr` не вимагається. `$(exec_prefix)/$(target_alias)` буде використовуватися, якщо система створена для крос-компілювання(для прикладу, компілюючи пакет на машині Intel, яка генерує код, який можу виконуватися на машинах PowerPC).

Важливо

Тестовий набір для Binutils у цій секції є критичним. Не пропускайте його виконання ні під якими обставинами.

Протестуйте результати:

```
make -k check
```

Встановіть пакет:

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

Встановіть заголовковий файл libiberty, який необхідний для деяких пакетів:

```
cp -v ../binutils-2.22/include/libiberty.h /usr/include
```

6.13.2. Вміст Binutils

Встановлені програми: addr2line, ar, as, c++filt, elfedit, gprof, ld, ld.bfd, nm, objcopy, objdump, ranlib, readelf, size, strings, and strip

Встановлені бібліотеки: libiberty.a, libbfd.{a,so}, and libopcodes.{a,so}

Встановлені директорії: /usr/lib/ldscripts

Короткий опис

addr2line	Перетворює програмні адреси у назви файлів і номери рядків; даючи адресу і назву виконуваного фалу, вона використовує відлагоджувальну інформацію у виконуваному файлі для визначення який файл джерела коду і номер рядка асоційований з адресою.
ar	Створює, модифікує і витягує з архівів
as	Асемблер, який обробляє вивід **gcc** у об'єктні файли
c++filt	Використовується компонувальником для фільтрування символів C++ і Java і стримує перевантажені функції від зіткнень
elfedit	Оновлює заголовкові файли ELF
gprof	Виводить так звану діаграму профільних даних
ld	Компонувальник, який комбінує множину об'єктних і архівних файлів у один файл, переміщуючи їхні дані і зв'язуючи символьні посилання
ld.bfd	Жорстке посилання до **ld**
nm	Перераховує символи, які є у даному об'єктному фалі
objcopy	Перетворює один тип об'єктного файлу у інший
objdump	Виводить інформацію про даний об'єктний файл, з опціями, які контролюють вивід необхідної інформації; ця інформація є корисною для програмістів, які працюють з інструментами компілювання

ranlib	Генерує індекс змісту архіву і зберігає його у архіві; індексний список усіх символів визначених членами архіву, які є переміщуваними об'єктними файлами
readelf	Виводить інформацію про тип бінарних файлів ELF
size	Перераховує розміри секцій і загальний розмір для даного об'єктного файлу
Strings	Виводить, для кожного даного їй файлу, послідовності друкованих символів, які мають специфіковану довжину (за замовчуванням до чотирьох); для об'єктних файлів, вона виводить, за замовчуванням, рядки, які є тільки з ініціалізуючих і завантажувальних секцій, а для інших типів файлів, вона сканує увесь файл
Strip	Видаляє символи відлагоджування з об'єктних файлів
`libiberty`	Вміщує підпрограми, які використовуються деякими програмами GNU, включаючи **getopt, obstack, strerror, strtol** і **strtoul**
`libbfd`	бібліотека Binary File Descriptor (дескриптор бінарних файлів)
`Libopcodes`	Бібліотека для роботи з опкодами — версія “здатного до прочитання тексту” інструкцій для процесору, використовуються для побудови утиліт, таких як **objdump**

6.14. GMP-5.0.5

Пакет GMP вміщує математичні бібліотеки. Вони мають корисні функції для арифметики з різною точністю.

Приблизний час побудови: 1.2 SBU
Необхідний дисковий простір: 50 Мбайт

6.14.1. Встановлення GMP

> Увага
>
> Якщо ви будуєте для 32-бітних систем архітектури x86, але ви маєте процесор здатний виконати 64-бітний код і ви встановили CFLAGS у середовищі, скрипт конфігурації буде намагатися конфігурувати для 64-бітів і провалиться. Зупиніть це викликаючи команду конфігурування з
>
> ```
> ABI=32 ./configure ...
> ```

Підготуйте GMP для компілювання:

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

Значення нових опцій конфігурування:

`--enable-cxx`

Цей параметер вмикає підтримку C++

`--enable-mpbsd`

Цей параметр будує бібліотеку сумісності Berkeley MP

Скомпілюйте пакет:

```
make
```

> Важливо
>
> Тестовий набір для GMP у цій секції є критичним. Не пропускайте його виконання ні під якими обставинами.

Протестуйте результати:

```
make check 2>&1 | tee gmp-check-log
```

Впевніться, що усі 166 тестів пройшли успішно. Перевірте результати, виконуючи наступну команду:

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

Встановіть пакет

```
make install
```

За бажанням, можна встановити документацію:

```
mkdir -v /usr/share/doc/gmp-5.0.5
cp -v doc/{isa_abi_headache,configuration} doc/*.html \
/usr/share/doc/gmp-5.0.5
```

6.14.2. Вміст GMP

Встановлені бібліотеки: libgmp.{a,so}, libgmpxx.{a,so}, and libmp.{a,so}

Встановлені каталоги: /usr/share/doc/gmp-5.0.5

Короткий опис

libgmp	Вміщує математичні функції точності
libgmpxx	Вміщує математичні функції точності для C++
libmp	Вміщує математичні функції Berkeley MP

6.15. MPFR-3.1.1

Пакет MPFR вміщує функції для математики з різною точністю

Приблизний час побудови: 0.8 SBU
Необхідний дисковий простір: 27 Мбайт

6.15.1. Встановлення MPFR

Підготуйте MPFR для компілювання:

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

Скомпілюйте пакет:

```
make
```

> Важливо
>
> Тестовий набір для MPFR у цій секції є критичним. Не пропускайте його виконання ні під якими обставинами.

Протестуйте результати і впевніться, що усі результати пройшли успішно:

```
make check
```

Встановіть пакет:

```
make install
```

Встановіть документацію

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

6.15.2. Вміст MPFR

Встановлені бібліотеки: libmpfr.{a,so}

Встановлені директорії: /usr/share/doc/mpfr-3.1.1

Короткий опис

`libmpfr` Містить математичні функції для роботи з різною точністю

6.16. MPC-1.0

Пакет MPC містить бібліотеку для арифметики комплексних чисел з довільно високою точністю і правильним округленням результату.

Приблизний час побудови: 0.4 SBU
Необхідний дисковий простір: 10.2 Мбайт

6.16.1. Встановлення MPC

Підготуйте MPC до встановлення:

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

Скомпілюйте пакет:

```
make
```

Для тестування результатів, виконайте:

```
make check
```

Встановіть пакет:

```
make install
```

6.16.2. Вміст MPC

Встановлені бібліотеки: libmpc.{a,so}

Короткий опис

libmpc	Містить математичні функції комплексних чисел

6.17. GCC-4.7.1

Пакет GCC містить колекцію компіляторів GNU, яка включає компілятори мов C і C++.

Приблизний час побудови: 53.5 SBU
Необхідний дисковий простір: 2.0 Гбайт

6.17.1. Встановлення GCC

Застосуйте заміну sed, що заборонить встановлення бібліотеки libiberty.a. Замість неї буде використовуватися версія libibery.a, яка постачається пакетом Binutils:

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

Як і у секції 5.9, “GCC-4.7.1 — прохід 2”, застосуйте команду sed щоб змусити побудову використовувати прапорець компілювання -fomit-frame-pointer, щоб забезпечити відповідну побудову компілятора:

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

Також, виправте помилку у одному з перевіряючих Makefile:

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

Документація GCC рекомендує побудову GCC ззовні дерева коду у спеціально-створеному каталозі:

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

Підготуйте GCC до компіляції:

```
../gcc-4.7.1/configure --prefix=/usr
     --libexecdir=/usr/lib
     --enable-shared
     --enable-threads=posix
     --enable-__cxa_atexit
     --enable-clocale=gnu
     --enable-languages=c,c++
     --disable-multilib
     --disable-bootstrap
     --with-system-zlib
```

Зауважте, що для інших мов програмування, є інші вимоги які тут не вказані. Читайте книгу BLFS для інструкцій побудови підтримки інших мов програмування пакету GCC.

Значення нових опцій конфігурування:

`--with-system-zlib`

Цей прапорець говорить GCC компонуватися з встановленою в системі копією бібліотеки Zlib ніж використовувати свою внутрішню копію.

Скомпілюйте пакет:

```
make
```

Важливо

Тестовий набір для MPFR у цій секції є критичним. Не пропускайте його виконання ні під якими

обставинами.

Відомо, що один з тестів у наборі тестів GCC вичерпує весь стек, отже зробіть більшим попередній розмір стеку, щоб виконати тести:

```
ulimit -s 32768
```

Протестуйте результати, але не зупиняйтеся через помилки:

```
make -k check
```

Для отримання результатів виконання тестів, виконайте:

```
../gcc-4.7.1/contrib/test_summary
```

Щоб отримати тільки результати тестів, профільтруйте вивід за допомогою команди **grep -A7 Summ.**

Результати можна порівняти з розміщеними з адресою *http://www.linuxfromscratch.org/lfs/build-logs/7.2/* і *http://gcc.gnu. org/ml/gcc-testresults/*.

Не можна уникнути деяких неочікуваних помилок. Розробники GCC зазвичай знають про ці проблеми, але поки-що їх не вирішують. Зокрема, відомо, що тести `libmudflap` є проблематичними які і результат помилки у GCC (*http://gcc.gnu.org/bugzilla/show_bug.cgi?id=20003*). Якщо результати тестів не сильно відрізняються від тих, що розміщені за вищезгаданою адресою, буде безпечно продовжувати.

Встановіть пакет:

```
make install
```

Деякі пакети, крім препроцесора C, будуть встановленими у директорію /lib. Щоб підтримувати ці пакети, створіть посилання:

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

Багато пакетів використовують **cc** щоб викликати компілятор C. Для того, щоб задовольнити ці пакети, створіть посилання:

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

Зараз, коли наш фінальний набір інструментів є на місці, важливо знову впевнитись, що компілювання і компонування буде працювати так, як очікується. Ми робимо це виконуючи перевірки, як ми і робили раніше:

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

Якщо все працює правильно, не повинно бути помилок і вивід останньої команди повинен бути (враховуючи специфічні для платформ назви динамічного компонувальника):

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

Зараз впевніться, що ми встановили використання коректних початкових файлів:

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

Якщо усе працює правильно, не повинно бути помилок і вивід останньої команди буде мати вигляд:

```
/usr/lib/gcc/i686-pc-linux-gnu/4.7.1/../../../crt1.o succeeded
/usr/lib/gcc/i686-pc-linux-gnu/4.7.1/../../../crti.o succeeded
/usr/lib/gcc/i686-pc-linux-gnu/4.7.1/../../../crtn.o succeeded
```

В залежності від архітектури нашої машини, вивід може трохи варіюватися, різниця зазвичай є у назві директорії після `/usr/lib/gcc`. Якщо ваша машина є 64-бітною, ви також можете бачити у цих рядках каталог, названий lib64. Важливою річчю, на яку ви маєте звертати увагу є те, що **gcc** повинен знайти усі файли `crt*.o` у директорії `/usr/lib`.

Перевірте чи компілятор шукає правильні заголовкові файли:

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

Команда повинна завершитись успішно, з наступним виводом:

```
#include <...> search starts here:
/usr/local/include
/usr/lib/gcc/i686-pc-linux-gnu/4.7.1/include
/usr/lib/gcc/i686-pc-linux-gnu/4.7.1/include-fixed
/usr/include
```

Майте на увазі, що директорія названа після вашої трійці букв платформи, може відрізнятися від вищезгаданої, в залежності від вашої архітектури.

> Увага
>
> Від версії 4.3.0, GCC беззастережно встановлює файл limits.h у приватну директорію included-fixed, і вимагається щоб ця директорія була на місці.

Наступним, перевірте чи компонувальник використовується з коректними шляхами пошуку:

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

Якщо усе працює коректно, не повинно бути ніяких помилок, і вивід останньої команди повинен бути приблизно таким:

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

На 64-бітних системах може бути дещо більше директорій. Для прикладу, ось вивід з машини архітектури x86_64:

```
SEARCH_DIR("/usr/x86_64-unknown-linux-gnu/lib64")
SEARCH_DIR("/usr/local/lib64")
SEARCH_DIR("/lib64")
SEARCH_DIR("/usr/lib64")
SEARCH_DIR("/usr/x86_64-unknown-linux-gnu/lib")
SEARCH_DIR("/usr/local/lib")
SEARCH_DIR("/lib")
SEARCH_DIR("/usr/lib");
```

Наступним, впевніться, що ми використовуємо коректну libc:

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

Якщо все працює коректно, не повинно бути ніяких помилок і вивід останньої команди буде:

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

Останнім чином, перевірте щоб GCC використовував коректний динамічний компонувальник:

```
grep found dummy.log
```

Якщо все працює коректно, не повинно бути помилок, і вивід останньої команди буде виглядати:

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

Якщо вивід який показаний вище не з'явився або не з'явився взагалі, в такому випадку щось пішло не так. Дослідіть і пройдіть усі кроки заново для того, щоб визначити де знаходиться проблема і виправте її. Найбільш ймовірною причиною неправильних дій, криється в відлагоджені функціональних файлів. Будь-яка проблема повинна бути вирішеною перед тим, як продовжувати процес.

Якщо все працює коректно, очистіть тестові файли:

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

Фінально, перемістіть файли, які не лежать на своєму місці:

```
mkdir -pv /usr/share/gdb/auto-load/usr/lib
mv -v /usr/lib/*gdb.py /usr/share/gdb/auto-load/usr/lib
```

6.17.2. Вміст GCC

Встановлені програми: c++, cc (link to gcc), cpp, g++, gcc, gccbug, and gcov

Встановлені бібліотеки: libgcc.a, libgcc_eh.a, libgcc_s.so, libgcov.a, libgomp.{a,so}, liblto_plugin.so, libmudflap.{a,so}, libmudflapth.{a,so}, libquadmath.{a,so}, libssp.{a,so}, libssp_nonshared.a, libstdc++.{a,so} and libsupc++.a

Встановлені директорії: /usr/include/c++, /usr/lib/gcc, /usr/share/gcc-4.7.1

Короткий опис

C++ Компілятор C++

cc Компілятор C

cpp Препроцесор C; він використовується компілятором, щоб обробити директиви #include, #define і інші подібні вирази у файлах коду

g++ Компілятор C++

gcc Компілятор C

gccbug Скрипт командної оболонки, який використовується для допомоги у створенні повідомлень про помилки

gcov Покриваюча тестова утиліта, вона використовується для аналізу програм, щоб визначити де оптимізацація буде мати найкращий ефект

`libgcc` Вміщує фукції **gcc**

`libgcov` Ця бібліотека компонується у програму, коли GCC проінструктований, щоб увімкнути профільвання

`libgomp` Реалізація GNU API OpenMP для багато-платформенного паралельного

	програмування з спільною пам'яттю у C/C++ і Fortran.
`liblto_plugin`	Плагін GCC`s Link Time Optimization (LTO) дозволяє GCC виконувати оптимізацію між одиницями компілювання
`libmudflap`	Вміщує підпрограми, які підтримують функціональність перевірки зв'язків GCC
`libquadmath`	Бібліотека GCC Quad Precision Math Library API (бібліотека API чотирикратної точності)
`libssp`	Містить підпрограми підтримки захисту руйнування цілісності стеку функцій GCC
`libstdc++`	Стандартна бібліотека C++
`libsupc++`	Забезпечує підпрограми підтримки для мови програмування C++

6.18. Sed-4.2.1

Пакет Sed містить редактор потоку

Приблизний час побудови: 0.2 SBU
Необхідний дисковий простір: 6.7 Мбайт

6.18.1. Встановлення Sed

Спочатку, виправте деякі тести:

```
patch -Np1 -i ../sed-4.2.1-testsuite_fixes-1.patch
```

Підготуйте Sed до компіляції:

```
./configure --prefix=/usr --bindir=/bin --htmldir=/usr/share/doc/sed-4.2.1
```

Значення нових опцій конфігурування:

--htmldir

Ця опція встановлює директорію де буде вміщуватися встановлена документація у форматі HTML.

Скомпілюйте пакет:

```
make
```

Згенеруйте документацію HTML:

```
make html
```

Для тестування результатів, виконайте:

```
make check
```

Встановіть пакет:

```
make install
```

Встановіть документацію HTML:

```
make -C doc install-html
```

6.18.2. Вміст Sed

Встановлена програма: sed

Встановлена директорія: /usr/share/doc/sed-4.2.1

Короткий опис

sed Фільтрує і перетворює текстові файли за один крок

6.19. Bzip-1.0.6

Пакет Bzip2 вміщує програми для компресії і декомпресії файлів. Компресія текстових файлів за допомогою bzip2 виконує набагато більший процент компресії ніж традиційний gzip.

Приблизний час побудови: менш ніж 0.1 SBU
Необхідний дисковий простір: 6.9 Мбайт

6.19.1. Встановлення Bzip2

Застосуйте патч, який встановить документацію для цього пакету:

```
patch -Np1 -i ../bzip2-1.0.6-install_docs-1.patch
```

Наступна команда запевнить встановлення символічних посилань

```
sed -i 's@\(ln -s -f \)$(PREFIX)/bin/@\1@' Makefile
```

Впевніться, що сторінки допомоги встановлені у правильне місце:

```
sed -i "s@(PREFIX)/man@(PREFIX)/share/man@g" Makefile
```

Підготуйте Bzip2 до компіляції:

```
make -f Makefile-libbz2_so
make clean
```

Значення параметрів make:

-f Makefile-libbz2_so

Це спричинить побудову Bzip2, використовуючи інший файл Makefile, у цьому випадку файл Makefile-libbz2_so, що створює динамічну бібліотеку libbz2.so і утиліти Bzip2 компонуються з нею.

Скомпілюйте і встановіть пакет:

```
make
```

Встановіть програму:

```
make PREFIX=/usr install
```

Встановіть динічний бінарний файл **bzip2** у каталог /bin, зробіть декілька необхідних посилань, і зробіть очистку:

```
cp -v bzip2-shared /bin/bzip2
cp -av libbz2.so* /lib
ln -sv ../../lib/libbz2.so.1.0 /usr/lib/libbz2.so
rm -v /usr/bin/{bunzip2,bzcat,bzip2}
ln -sv bzip2 /bin/bunzip2
ln -sv bzip2 /bin/bzcat
```

6.19.2. Вміст Bzip2

Встановлені програми: bunzip2 (link to bzip2), bzcat (link to bzip2), bzcmp (link to bzdiff), bzdiff, bzegrep (link to bzgrep), bzfgrep (link to bzgrep), bzgrep, bzip2, bzip2recover, bzless (link to bzmore), and bzmore

Встановлені бібліотеки: libbz2.{a,so}

Встанаовлені директорії: /usr/share/doc/bzip2-1.0.6

Короткий опис

bunzip	Декомпресія файлів bzip2
bzcat	Декомпресія в стандартний потік виводу
bzcmp	Виконує **cmp** на стиснених файлах
bzdiff	Виконує **diff** на стиснених файлах
bzegrep	Виконує **egrep** на стиснених файлах
bzfgrep	Виконує **fgrep** на стиснених файлах
bzgrep	Виконує **grep** на стиснених файлах
bzip2	Стискає файли, використовуючи стиснення блоковим текстовим сортуванням методом алгоритму Бровс-Вілера (Burrows-Wheeler) з кодуванням Хуфмана (Huffman); швидкість стиснення є кращою ніж отримана більш звичними компресорами, які використовують алгоритм “Лемпел-Зіва” (“Lemple-Ziv”), такі як gzip
bzip2recover	Намагається відновити дані з пошкоджених файлів bzip2
bzless	Виконує **less** на файлах bzip2
bzmore	Виконує **more** на файлах bzip2
`libbz2*`	Бібліотека, яка реалізує не витратну, компресію блочного сортування даних, використовуючи алгоритм Буровс-Вілера (Burrows-Wheeler)

6.20. Pkg-config-0.27

Пакет pkg-config містить утиліту для проходження шляхів підключення і/або шляхів до бібліотек для побудови інструментів під час конфігурації і виконання файлів make.

Приблизний час побудови: 0.4 SBU
Необхідний дисковий простір: 30 Мбайт

6.20.1. Встановлення Pkg-config

Підготуйте Pkg-config до компіляції:

```
./configure --prefix=/usr \
--with-internal-glib \
--docdir=/usr/share/doc/pkg-config-0.27
```

Скомпілюйте пакет:

```
make
```

Для проходження тестів, виконайте:

```
make check
```

Встановіть пакет:

```
make install
```

6.20.2. Вміст Pkg-config

Встановлені програми: pkg-config

Встановлені директорії: /usr/share/doc/pkg-config-0.26

Короткий опис

pkg-config Повертає мета інформацію для даного пакету чи бібліотеки

6.21. Ncurses-5.9

Пакет Ncurses вміщує бібліотеки для термінально-незалежної обробки символів екрану

Приблизний час побудови: 0.6 SBU
Необхідний дисковий простір: 40 Мбайт

6.21.1. Встановлення Ncurses

Підготуйте Ncurses до куомпіляції:

```
./configure --prefix=/usr --mandir=/usr/share/man --with-shared \
--without-debug —enable-widec
```

Значення опцій конфігурування:

--enable-widec

Цей прапорець спричиняє побудову широко-символьних бібліотек (таких як `libncursesw.so.5.9`) замість нормальних (`libncurses.so.5.9`). Ці широко-символьні бібліотеки корисні для використання з багато байтними і традиційними 8-бітними локалями, коли нормальні бібліотеки працюють правильно тільки з 8-бітними локалями. Широко-символьні і нормальні бібліотеки є сумісними з вихідним кодом, але не не сумісним по бінарному інтерфейсу.

Скомпілюйте пакет:

```
make
```

Цей пакет має тестовий набір, але він може бути виконаним після того, як пакет буде встановлено. Тести розміщуються у директорії test/. Читайте файл README у цій директорії для більшої інформації.

Встановіть пакет:

```
make install
```

Перемістіть динамічні бібліотеки у директорію /lib, де вони очікуються бути:

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

Через те, що бібліотеки були переміщеними, одне посилання вказуватиме на неіснуючий файл. Створіть його знову:

```
ln -sfv ../../lib/libncursesw.so.5 /usr/lib/libncursesw.so
```

Багато програм все одно очікують, щоб компонувальник знаходив не широко-символьну бібліотеку Ncurses. Обманіть такі програми компонуванням з широко-символьною бібліотекою посиланнями і скриптами компонувальника:

```
for lib in ncurses form panel menu ; do \
    rm -vf /usr/lib/lib${lib}.so ; \
    echo "INPUT(-l${lib}w)" >/usr/lib/lib${lib}.so ; \
    ln -sfv lib${lib}w.a /usr/lib/lib${lib}.a ; \
done
ln -sfv libncurses++w.a /usr/lib/libncurses++.a
```

Фінально, впевніться, що старі програми, які шукають прапорець -lcurses в час побудови, можуть бути побудованими:

```
rm -vf /usr/lib/libcursesw.so
echo "INPUT(-lncursesw)" >/usr/lib/libcursesw.so
```

```
ln -sfv libncurses.so /usr/lib/libcurses.so
ln -sfv libncursesw.a /usr/lib/libcursesw.a
ln -sfv libncurses.a /usr/lib/libcurses.a
```

За бажанням, встановіть документацію Ncurses:

```
mkdir -v /usr/share/doc/ncurses-5.9
cp -v -R doc/* /usr/share/doc/ncurses-5.9
```

Увага

Інструкції розміщені вище не створюють звичайні бібліотеки Ncurses , так як ніякі пакети встановлені компілюванням з джерел коду будуть компонуватися з ними в час виконання. Якщо ви маєте мати такі бібліотеки через деякі бінарні програми, або щоб бути сумісними з LSB, побудуйте пакет знову з наступними командами:

```
make distclean
./configure --prefix=/usr --with-shared --without-normal \
--without-debug --without-cxx-binding
make sources libs
cp -av lib/lib*.so.5* /usr/lib
```

6.21.2. Вміст Ncueses

Встановлені програми: captoinfo (link to tic), clear, infocmp, infotocap (link to tic), ncursesw5-config, reset (link to tset), tabs, tic, toe, tput, and tset

Встановлені бібліотеки: libcursesw.{a,so} (посилання і компонувальний скрипт до libncursesw.{a,so}), libformw.{a,so}, libmenuw.{a,so}, libncurses++w.a, libncursesw.{a,so}, libpanelw.{a,so} і їхні не широко-символьні частини без "w" у іменах бібліотек.

Встановлені директорії: /usr/share/tabset, /usr/share/terminfo

Короткий опис

captinfo Конвертує опис termcap у опис terminfo

clear Очищає екран, якщо це можливо

infocmp Порівнює або виводить опис terminfo

infotocap Конвертує опис terminfo у опис termcap

ncursesw5-config Заюезпечує інформацію конфігурації для ncurses

reset Реініціалізує термінал до його значень за замовчуванням

tabs Очищає і встановлює зупинки tab в терміналі

tic Компілятор вступних описів terminfo, що перетворює файли terminfo з формату джерела у бінарний формат, який необхідний для підпрограм бібліотек ncurses. Файл terminfo містить інформацію про можливості

	певних терміналів.
toe	Перераховує усі можливі типи терміналів, даючи головне ім'я і опис для кожного
tput	Робить значення термінально-залежних можливостей доступними для командних оболонок; вона також можу бути використаною, щоб збити чи ініціалізувати термінал чи вивести його довге ім'я
tset	Можу бути використаний для ініціалізування терміналів
`libcurses`	посилання до libncurses
`libncurses`	Містить функції, для представлення тексту у багатьох комплексних шляхах на терміналі екрану; хороший приклад використання цих функцій є меню, яку виводиться під час програми ядра **make menuconfig**
`libfrom`	Вміщує функції для реалізації форм
`libmenu`	Містить функції для реалізації меню
`libpanel`	Містить функції для реалізації панелей

6.22. Util-linux-2.21.2

Пакет Util-linux містить різні утилітні програми. Серед них є утиліти для обробки файлових систем, консолей, розділів і повідомлень.

Приблизний час побудови: 0.7 SBU
Необхідний дисковий простір: 81 Мбайт

6.22.1. Нотатки дотримання FHS

FHS рекомендує використовувати каталог /vat/lib/hwclock замість /etc для розміщення файлів adjtime. Щоб зробити програму hwclock сумісну з FHS, виконайте наступне:

```
sed -i -e 's@etc/adjtime@var/lib/hwclock/adjtime@g' \
$(grep -rl '/etc/adjtime' .)
mkdir -pv /var/lib/hwclock
```

6.22.2. Встановлення Util-linux

```
./configure
```

Скомпілюйте пакет

```
make
```

Цей пакет не постачається з тестовим набором.

Встановіть пакет:

```
make install
```

6.22.3. Вміст Util-linux

Встановлені програми: addpart, agetty, blkid, blockdev, cal, cfdisk, chcpu, chkdupexe, chrt, col, colcrt, colrm, column, ctrlaltdel, cytune, delpart, dmesg, fallocate, fdformat, fdisk, findfs, findmnt, flock, fsck, fsck.cramfs, fsck.minix, fsfreeze, fstrim, getopt, hexdump, hwclock, i386, ionice, ipcmk, ipcrm, ipcs, isosize, ldattach, linux32, linux64, logger, look, losetup, lsblk, lscpu, mcookie, mkfs, mkfs.bfs, mkfs.cramfs, mkfs.minix, mkswap, more, mount, mountpoint, namei, partx, pg, pivot_root, prlimit, raw, readprofile, rename, renice, rev, rtcwake, script, scriptreplay, setarch, setsid, setterm, sfdisk, swaplabel, swapoff (link to swapon), swapon, switch_root, tailf, taskset, tunelp, ul, umount, unshare, uuidd, uuidgen, wall, whereis, wipefs, and x86_64

Встановлені бібліотеки: libblkid.{a,so}, libmount.{a,so}, libuuid.{a,so}

Встановлені директорії: /usr/include/blkid, /usr/include/libmount, /usr/include/uuid, /usr/share/getopt, /var/lib/hwclock

Короткий опис

addpart Повідомляє ядро Лінукс про нові розділи

agetty Відриває порт tty, підказує ввід ім'я логіну, і тоді запускає програму **login**

blkid	Утиліта командної оболонки, яка використовується для встановлення і виводити атрибути блокового пристрою
blockdev	Дозволяє користувачам викликати кластери вводу виводу для блокових пристроїв з командної оболонки
cal	Виводить простий календар
cfdisk	Маніпулює таблицею розділів даного пристрою
chcpu	Модифікує стани процесора
chkdupexe	знаходить дублювані виконувані файли
chrt	Маніпулює атрибути процесу під час виконання
col	Фільтрує канали реверсних рядків
colcrt	Фільтрує вивід **nroff** для терміналів, які не мають деяких можливостей
colrm	Фільтрує дані колонки
column	Форматує даний файл у кілька колонок
ctrlaltdel	Встановлює функцію для комбінації клавіш Ctrl+Alt+Del до апаратного чи програмного перевантаження
cytune	Редагує параметри драйверів серійних ліній для карт Cyclades
delpart	Дає запит до ядра Лінукс про видалення розділу
dmesg	Показує повідомлення завантаження ядра
fallocate	Займає місце для файлу
fdformat	Низькорівневе форматування дискет
fdisk	Маніпулює таблицею розділів даного пристрою
findfs	Знаходить файлову систему за міткою Universally Unique Identifier (UUID)
findmnt	Це командний інтерфейс до бібліотеки libmount для роботи з інформацією про монтування, файлів fstab і mtab
flock	Одержує блокування файлу і тоді виконує команду з блокуванням
fsck	Використовується для перевірки, і опціонального ремонтування, файлових систем

fsck.cramfs	Виконує узгоджену перевірку на файловій системі Cramfs даного пристрою
fsck.minix	Виконує узгоджену перевірку на файловій системі Minix даного пристрою
fsfreeze	Проста обгортка для ioctl FIFREEZE/FITHAW операцій з драйверами ядра
fstrim	Відкидає не використовувані блоки пристрою монтованої файлової системи
getopt	Розбирає опції у даному командному рядку
hexdump	Виводить даний їй файл у шістнадцятковому чи іншому форматі
hwclock	Читає ч встановлює апаратний системний годинник, також названий Годинник Реального Часу (Real-Time Clock RTC) чи годинник системи BIOS (Basic Input-Output System).
i386	Символічне посилання на setarch
ionice	Повертає чи встановлює класи схем введення/виведення і пріоритетність для програми
ipcmk	Створює деякі ресурси IPC
ipcrm	Видаляє даний їй ресурс міжпроцесної взаємодії (Inter-Process Communication — IPC)
ipcs	Надає інформацію статусу IPC
isosize	Повідомляє розмір файлової системи iso9660
ldattach	Приєднує рядок до серійного
linux32	Символічне посилання до setatch
linux64	Символічне посилання до setatch
logger	Записує дане повіддомлення до системного журналу
look	Виводить рядки які починаються з даної стрічки
losetup	Встановлює і контролює кругові пристрої (loop devices)
lsblk	Виводить інформацію про усі чи вибрані блокові пристрої у деревоподібному форматі
lscpu	Виводить інформацію про архітектуру процесора
mcookie	Генерує магічні cookie (128-бітне випадкове шістнадцяткове число) для **xauth**

mkfs	Будує файлову систему на пристрої (зазвичай розділ жорсткого диску)
mkfs.bfs	Створює файлову систему Santa Cruz Operations (SCO)
mkfs.cramfs	Створює файлову систему cramfs
mkfs.minix	Створю файлову систему minix
mkswap	Ініціалізує даний розділ чи файл для використання як області підкачки
more	Фільтер для виводу сторінок на екран
mount	Приєднує файлову систему на даному пристрої до виділеної директорії у дереві файлової системи
mountpoint	Перевіряє чи директорія є точкою монтування
namei	Показує символічні посилання у даних шляхів
partx	Каже ядру про присутність і нумерування розділів на диску
pg	Виводить текстовий файл повністю на весь екран
pivot_root	Робить дану файлову систему новою кореневою системою для поточних процесів
prlimit	Встановлює і дістає ресурсні обмеження процесів
raw	Зв'язує сирий символьний пристрій Лінукса до блокового пристрою
readprofile	Читає профільну інформацію ядра
rename	Перейменовує дані файли, заміняючи дану строку іншою
renice	Змінює пріоритет виконуваних процесів
rev	Робить зворотній порядок рядків даного файлу
rtcwake	Використовується для вводу системи в режим сну поки не настане певний заданий час
script	Робить машинопис термінальної сесії
scriptreplay	Відтворює машинопис використовуючи інформацію про час
setarch	Змінює виведену архітектуру у новому програмному середовищі і встановлює особливі прапорці
setsid	Виконує дану програму у новій сесії

setterm	Встановлює атрибути терміналу
sfdisk	Маніпулятор таблиці розділів
swaplabel	Дозволяє змінювати UUID і мітку області swap
swapoff	Вимикає пристрої і файли для свопінгу
swapon	Вмикає пристрої і файли для свопінгу і виводить пристрої і файли які використовуються в даний час
switch_root	Перемикає до іншої файлової системи як кореневу точку монтування дерева
tailf	Показує зростання файлу журналу. Виводить принаймні 10 рядків файлу журналу, потім, продовжує виводити будь-які нові записи у файлі
taskset	Дістає або встановлюємо спорідненість процесу з процесором
tunelp	Змінює параметри друкованих рядків
ul	Фільтр для заміни символів підкреслювання у керувальні послідовності вказуючи підкреслення використовуваного терміну
unmount	Відключає файлову систему від кореневого дерева каталогів
unshare	Виконує програму з деяким простором символів, який невидимий для батьківського процесу
uuidd	Демон, який використовується бібліотекою UUID для генерації основаних на часі UUID у безпечному і гарантовано унікальному стилі
uuidgen	Створює нові UUID. Кожен новий UUID може не безпричинно вважатися унікальним серед усіх створених UUID, на локальній системі і на інших системах, у минулому і майбутньому
wall	Виводить контент файлу чи, за замовчуванням, його стандартний ввід, на термінали усіх присутніх користувачів
whereis	Виводить розміщення бінарних, файлів коду і сторінок допомоги для даної команди
wipefs	Стирає сигнатуру файлової системи з пристрою
x86_64	Символічне посилання до setarch
libblkid	Містить підпрограми для ідентифікації пристрою і маркер видобутку
libuuid	Містить програми для генерації унікальних ідентифікаторів для об'єктів, які можуть бути доступними за межами локальної системи

6.23. Psmisc-22.19

Пакет Psmisc вміщує програми для вивожу інформації про виконувані процеси

Приблизний час побудови: менш ніж 0.1 SBU
Необхідний дисковий простір: 4.2 Мбайт

6.23.1. Встановлення Psmisc

Підготуйте Psmisc до компіляції

```
./configure —prefix=/usr
```

Скомпілюйте пакет

```
make
```

Цей пакет не постачається з тестовими наборами

Встановіть пакет:

```
make install
```

Фінально, перемістіть програми killall і fuser до місце специфікованого стандартом FHS

```
mv -v /usr/bin/fuser /bin
mv -v /usr/bin/killall /bin
```

6.23.2. Вміст Psmisc

Встановлені програми: fuser, killall, peekfd, prtstat, pstree, і pstree.x11 (посилання до pstree)

Короткий опис

fuser Виводить ID процесів (PID) процесів, які використовують даний файл чи файлову систему

killall Вбиває процеси за їх ім'ям; вона посилає сигнал до усіх працюючих процесів усім даним командам

peekfd Переглядає файлові дескриптори працюючих процесів, за їхніми PID

prtstat Виводить інформацію про процеси

pstree Виводить виконувані процеси у вигляді дерева

pstree.x11 Так само як і **pstree** крім того, що вона чекає підтвердження перед виходом

6.24. E2fsprogs-1.42.5

Пакет E2fsprogs вміщує утиліти для обробки файлової системи ext2. Він також підтримує журнальні файлові системи ext3 і ext4.

Приблизний час побудови: 1.7 SBU
Необхідний дисковий простір: 64 Мбайт

6.24.1. Встановлення E2fsprogs

Документація E2fsprogs рекомендує, щоб пакет будувався в окремій спеціальній субдиректорії дерева коду:

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

Підготуйте E2fsprogs до компіляції

```
../configure --prefix=/usr \
--with-root-prefix="" \
--enable-elf-shlibs \
--disable-libblkid \
--disable-libuuid \
--disable-uuidd \
--disable-fsck
```

Значення опцій конфігурування:

`--with-root-prefix=""`

Деякі програми (такі як **e2fsck**) вважаються істотними. Коли, для прикладу, `/usr` не монтована, ці програми все одно повинні бути доступними. Вони належать до таких директорій як `/lib` і `/sbin`. Якщо ця опція не дана для скрипту конфігурації E2fsprogs, ці програми встановлюються у директорію `/usr`.

`--enable-elf-shlibs`

Це створює динамічні бібліотеки, які використовують деякі програми

`--disable-*`

Це запобігає E2fsprogs від будування і встановлення бібліотек `libuuid` і `libblkid`, демон uuidd і обгортка fsck, так як вони були встановлені раніше за допомогою Util-Linux.

Скомпілюйте пакет:

```
make
```

Для тестування результатів:

```
make -k check
```

Один з тестів E2fsprogs буде намагатися виділити 256 Мбайт пам'яті. Якщо ви не маєте більше RAM пам'яті, ніж вимагає тест, рекомендується ввімкнути достатній розмір swap для тестів. Дивіться Секцію 2.3, “Створення файлової системи на розділі” і Секцію 2.4, “Монтування нового розділу” для більшої інформації і вмикання області swap.

Встановіть бінарні файли, документацію і динамічні бібліотеки:

```
make install
```

Встановіть статичні бібліотеки і заголовкові файли:

```
make insstall-libs
```

Зробіть встановлені статичні бібліотеки доступними до запису, щоб символи відлагодження могли бути видаленими пізніше:

```
chmod -v u+w /usr/lib/{libcom_err,libe2p,libext2fs,libss}.a
```

Цей пакет встановлює стиснені за допомогою gzip файли .info, але не оновлюйте загальносистемний файл dir. Розпакуйте цей файл і тоді оновіть системний файл dir, використовуючи наступні команди:

```
gunzip -v /usr/share/info/libext2fs.info.gz
install-info —dir-file=/usr/share/info/dir /usr/share/info/libext2fs.info
```

За бажанням, створіть і встановіть деяку додаткову інформацію використовуючи наступні команди:

```
makeinfo -o doc/com_err.info ../lib/et/com_err.texinfo
install -v -m644 doc/com_err.info /usr/share/info
install-info --dir-file=/usr/share/info/dir /usr/share/info/com_err.info
```

6.24.2. Вміст E2fsprogs

Встановлені програми:	badblocks, chattr, compile_et, debugfs, dumpe2fs, e2freefrag, e2fsck, e2image, e2initrd_helper, e2label, e2undo, e4defrag, filefrag, fsck.ext2, fsck.ext3, fsck.ext4, fsck.ext4dev, logsave, lsattr, mk_cmds, mke2fs, mkfs.ext2, mkfs.ext3, mkfs.ext4, mkfs.ext4dev, mklost+found, resize2fs і tune2fs
Встановлені бібліотеки:	libcom_err.{a,so}, libe2p.{a,so}, libext2fs.{a,so}, libquota.a and libss.{a,so}
Встановлені директорії:	/usr/include/e2p, /usr/include/et, /usr/include/ext2fs, /usr/include/quota, /usr/include/ss, /usr/share/et, /usr/share/ss

Короткий опис

badblocks	Робить пошук (зазвичай на зозділі) пошкоджених блоків
chattr	Змінює атрибути для файлів на файловій системі ext2; вона також змінює їх у файловій системі ext3 — журнальну версію файлової системи ext2.
compile_et	Компілятор таблиць помилок; він перетворює таблицю кодів помилок і їх опис у файл коду мови C, який підходить для використання з бібліотекою com_err.
Debugfs	Налагоджувач файлової системи; він може бути використаним для перевірки і зміни стану файлової системи ext2
dumpe2fs	Виводить інформацію суперблоку і групи блоків для файлової системи на даному пристрої
e2freeflag	Виводить інформацію про вільний фрагментований об'єм диску
e2fsck	Використовується для перевірки і опціонального ремонту файлової системи ext2 і ext3

e2image	Використовується для збереження критичних даних файлової системи ext2 у файл
e2initrd_helper	Виводить ти файлової системи даного розділу, по даній мітці чи по ім'я
e2label	виводить чи змінює мітку файлової системи на файловій системі ext2
e2undo	Робить повернення дій за допомогою файлу журналу undo_log для для файлових систем ext2/ext3/ext4 знайдених на пристрої. Це може бути використано, щоб повернути провалені операції виконані програмами e2fsprogs
e4defrag	Дефрагментатор для систем ext4
filefrag	Виводить на скільки файл може бути фрагментований
fsck.ext2	За замовчуванням перевіряє файлову систему ext2. Це є жорстке посилання до e2fsck
fsck.ext3	За замовчуванням перевіряє файлову систему ext3. Це є жорстке посилання до e2fsck
fsck.ext4	За замовчуванням перевіряє файлову систему ext4. Це є жорстке посилання до e2fsck
fsck.ext4dev	За замовчуванням перевіряє розроблювані файлові системи ext4. Це є жорстке посилання до e2fsck
logsave	Зберігає вивід команди у файлі журналу
lsattr	Виводить атрибути файлів на другі розширеній файловій системі
mk_cmds	Перетворює таблицю імен команд і повідомлень допомоги у файл коду мови C для використання підсистемою бібліотеки `libss`
mke2fs	Створює файлові системи ext2 чи ext3 на даному пристрої
mkfs.ext2	За замовчуванням створює файлову систему ext2. Це є жорстку посилання до **mke2fs**.
mkfs.ext3	За замовчуванням створює файлову систему ext3. Це є жорстку посилання до **mke2fs**.
mkfs.ext4	За замовчуванням створює файлову систему ext4. Це є жорстку посилання до **mke2fs**.
mkfs.ext4dev	За замовчуванням створює розроблювану файлову систему ext4. Це є жорстке посилання до **mke2fs**

mklost+found	Використовується для створення директорії lost+found на файловій системі ext2; вона займає дискові блоки до цієї директорії для полегшення завдання команди e2fsck
resize2fs	Може бути використаною для збільшення чи зменшення файлових систем ext2
tune2fs	Коригує параметри файлові системи тип якої це дозволяє
`libcom_err`	Поширена підпрограма виводу помилок
`libe2p`	Використовується програмами **dumpe2fs, chattr** і **lsattr**
`libquota`	Постачає інтерфейс для створення і оновлення файлів квот і полів суперблоків ext4
`libss`	Використовується **debugfs**

6.25. Shadow-4.1.5.1

Пакет Shadow містить програми для обробки паролів безпечним шляхом.

Приблизний час побудови: 0.2 SBU
Необхідний дисковий простір: 42 Мбайт

6.25.1. Встановлення Shadow

> Увага
>
> Якщо б ви хотіли дотримуватися використання жорстких паролів, зверніться до *http://www.linuxfromscratch.org/blfs/view/svn/postlfs/cracklib.html* для встановлення CrackLib, який важливо встановити першим перед Shadow. Тоді добавте `--with-libcrack` до наступної команди **configure**.

Вимкніть встановлення програми groups і її сторінок документації, так як Coreutils постачають кращі версії:

```
sed -i 's/groups$(EXEEXT) //' src/Makefile.in
find man -name Makefile.in -exec sed -i 's/groups\.1 / /' {} \;
```

Замість того, щоб використовувати звичний метод crypt, використовуйте більш безпечний SHA-512 метод криптування паролів, який також дозволяє застосовувати паролі довші ніж 8 символів. Також необхідно змінити абсолютне місце знаходження користувацьких поштових скриньок за /var/spool/mail, які Shadow використовує за замовчуванням до шляху `/var/mail`:

```
sed -i -e 's@#ENCRYPT_METHOD DES@ENCRYPT_METHOD SHA512@' \
-e 's@/var/spool/mail@/var/mail@' etc/login.defs
```

> Увага
>
> Якщо ви використовуєте підтримку CrackLib для побудови Shadow, виконайте наступне:
>
> ```
> sed -i 's@DICTPATH.*@DICTPATH\t/lib/cracklib/pw_dict@' \
> etc/login.defs
> ```

Підготуйте Shadow до компілювання:

```
./configure --sysconfdir=/etc
```

Скомпілюйте пакет:

```
make
```

Цей пакет не постачається з тестовим набором програм.

Встановіть пакет:

```
make install
```

Перемістіть програму на її правильне місце:

```
mv -v /usr/bin/passwd /bin
```

6.25.2.Конфігурування Shadow

Цей пакет вміщує утиліти для додавання, модифікування і видалення користувачів і груп; встановлювати і змінювати їхні паролі; і виконувати інші адміністративні завдання. Для повного роз'яснення, що саме означає

passwdord shadowing, прочитайте файл doc/HOWTO у розпакованому дереві вихідного коду. Якщо використовувати підтримку Shadow, майте на увазі, що програми які потребують перевірку паролей (віконні менеджери, програми для роботи з FTP, pop3 демони чи інші) мають бути сумісними з Shadow. Це означає, що вони мають вміти працювати з прихованими паролями.

Для увімкнення прихованих паролей, виконайте наступну команду:

```
pwconv
```

Для ввімкнення прихованих паролів груп, виконайте:

```
grpconv
```

Стокова конфігурація для утиліти **useradd** має декілька застережень, які потребують деякого роз'яснення. Для початку, звичайною лією для утиліти **useradd** буде створення користувача і групу, яка буде мати таке саме ім'я як і користувач. Зазвичай числа ID користувача (UID) і ID групи (UID) починаються з 1000. Це означає, якщо ви не даєте параметри до **useradd,** тоді кожен користувач буде членом унікальної групи на системі. Якщо ця поведінка є небажаною, вам необхідно дати параметер -g до **useradd.** Параметри за умовчанням зберігаються у файлі /etc/default/useradd. Вам може бути необхідно модифікувати два параметри у цьому файлі для вирішення ваших приватних проблем.

Роз'яснення параметрів файлу /etc/default/useradd

GROUP=1000

Цей параметр встановлює початок номерів груп, які використовуються у файлі /etc/group. Ви можете модифікувати його за любим вашим бажанням. Майте на увазі, що **useradd** не буде повторно використовувати UID чи GID. Якщо число ідентифіковане як використане, вона буде використовувати наступне вільне число після нього. Зауважте також, якщо ви не маєте групу 1000 на вашій системі, вперше, коли ви використаєте **useradd** без параметра -g, ви дістанете вивід повідомлення яке каже наступне: useradd: unknown GID 1000. Ви також можете нехтувати цим повідомленням і група під номером 1000 буде використаною.

CREATE_MAIL_SPOOL=yes

Цей параметр примушує **useradd** створити файл поштової скриньки для новоствореного користувача. **Useradd** також зробить власником цього файлу групу mail з правами 0660. Якщо ви надаєте перевагу, щоб **useradd** не створював ці файли mailbox, виконайте наступну команду:

```
sed -i 's/yes/no/' /etc/default/useradd
```

6.25.3. Встановлення пароля користувача root

Виберіть пароль для користувача root і встановіть його виконуючи:

```
passwd root
```

6.26.4. Вміст Shadow

Встановлені програми:	chage, chfn, chgpasswd, chpasswd, chsh, expiry, faillog, gpasswd, groupadd, groupdel, groupmems, groupmod, grpck, grpconv, grpunconv, lastlog, login, logoutd, newgrp, newusers, nologin, passwd, pwck, pwconv, pwunconv, sg (link to newgrp), su, useradd, userdel, usermod, vigr (link to vipw), and vipw
Встановлені директорії:	/etc/default

Короткий опис

chage	Використовується для зміни максимального числа днів між обов'язковими змінами паролів
chfn	Використовується для зміни повного ім'я користувача і іншої інформації
chgpasswd	Використовується для оновлення паролю групи
chpasswd	Використовується для зміни паролю користувача
chsh	Використовується для заміни командної звичної оболонки користувача
expiry	Перевіряє і забезпечує політику витікання паролів
faillog	Використовується для перевірки журналу провалів входу у систему, встановлення максимального числа помилок авторизації перед тим, як профіль буде заблоковано, чи збити обрахунок помилок авторизації
gpasswd	Використовується для додання і видаляння членів і адміністраторів групи
groupadd	Створює групу з даним ім'ям
groupdel	Видаляє групу з даним ім'ям
groupmems	Дозволяє користувачу адмініструвати його членство в його групі без вимоги привілеїв суперкористувача
groupmod	Використовується для модифікування даного ім'я групи чи GID
grpck	Перевіряє цілісність файлів груп `/etc/group` і `/etc/shadow`
grpconv	Створює і оновлює файли груп shadow з нормально файлу груп
grpunconv	Оновлює `/etc/group` з `/etc/gshadow` і видаляє останній
lastlog	Виводить останню авторизацію користувачів чи даного користувача
login	Використовується системою для авторизації користувачів
logoutd	Демон, який використовується для забезпечення обмежень під час присутності в системі
newgrp	Використовується для заміни поточного GID групи під час сесії входу
newusers	Використовується для створення і оновлення цілої серії профілів користувачів
nologin	Виводить повідомлення про те що профіль є недоступним. Створено для використання як оболонка за замовчуванням для профілів, які були

	вимкнені
passwd	Використовується для зміни паролю для профілю користувача чи групи
pwck	Перевіряє цілісність файлів паролів `/etc/passwd` і `/etc/shadow`
pwconv	Створює чи оновлює пароль файлу shadow з нормального файлу паролів
pwunconv	Оновлює `/etc/passwd` з `/etc/shadow` і тоді видаляє останній
sg	Виконує дану команду поки GID користувача встановлено до певної групи
su	Запускає командну оболонку з заміною ID користувача і групи
useradd	Створюж нового користувача з даним ім'ям, чи оновлює звичну інформацію про користувача
userdel	Видалює даний профіль користувача
usermod	Використовується для модифікування даного ім'я користувача, ідентифікатор користувача (ID), командну оболонку, початкову групу, домашній каталог і інше
vigr	Редагує файли `/etc/group` чи `/etc/gshadow`
vipw	Редагує файли `/etc/passwd` чи `/etc/shadow`

6.26. Coreutils-8.19

Пакет Coreutils містить утиліти для показу і встановлення базових характеристик системи.

Приблизний час побудови: 4.0 SBU
Необхідний дисковий простір: 154 Мбайт

6.26.1. Встановлення Coreutils

Виправте помилку у тестах:

```
sed -i -e 's/! isatty/isatty/' \
-e '45i\ || errno == ENOENT' gnulib-tests/test-getlogin.c
```

Стандарт POSIX вимагає, щоб програми з пакету Coreutils коректно розпізнавали символи, навіть у мультибайтових локалей. Наступний патч виправляє це недотримання і інші помилки залежні від інтернаціоналізацій:

```
patch -Np1 -i ../coreutils-8.19-i18n-1.patch
```

> Увага
>
> У минулому, багато помилок знаходились в даному патчі. Перш ніж оголошувати нову помилку до постачальників Coreutils, будь-ласка, перевірте спочатку чи вони репродукуються без даного патчу.

Тепер підготуйте Coreutils до компіляції

```
FORCE_UNSAFE_CONFIGURE=1 ./configure \
--prefix=/usr \
--libexecdir=/usr/lib \
--enable-no-install-program=kill,uptime
```

Значення опцій конфігурування:

`--enable-no-install-program=kill,uptime`

Ціль цього прапорця у тому, щоб запобігти Coreutils встановити бінрні файли, які будуть встановленими іншими пакетами пізніше.

Скомпілюйте пакет:

```
make
```

Пропустіть “Встановлення пакету”, якщо не виконано тестових прогшрам.

Зараз тестовий набір готовий до виконання. Для початку, виконайте тест, який має виконуватися під користувачем root:

```
make NON_ROOT_USERNAME=nobody check-root
```

Ми будемо виконувати вищезгадані тести як користувач nobody. Деякі тести, однак, вимагають, щоб користувач був членом більш ніж однієї групи. Отож, якщо ці тести не були пропущеними, ми додамо тимчасову групу і зробимо користувача nobody частиною її:

```
echo "dummy:x:1000:nobody" >> /etc/group
```

Виправте деякі права доступу так, щоб відмінні від root користувачі могли компілювати і виконувати тести:

```
chown -Rv nobody
```

Тепер, виконайте тести. Впевніться, що змінна `PATH` у середовищі **su** містить `/tools/bin`.

```
su nobody -s /bin/bash \
-c "PATH=$PATH make RUN_EXPENSIVE_TESTS=yes -k check || true"
```

Видаліть тимчасову групу:

```
sed -i '/dummy/d' /etc/group
```

Встановіть пакет:

```
make install
```

Перемістіть програми у місця специфіковані стандартом FHS:

```
mv -v /usr/bin/{cat,chgrp,chmod,chown,cp,date,dd,df,echo} /bin
mv -v /usr/bin/{false,ln,ls,mkdir,mknod,mv,pwd,rm} /bin
mv -v /usr/bin/{rmdir,stty,sync,true,uname} /bin
mv -v /usr/bin/chroot /usr/sbin
mv -v /usr/share/man/man1/chroot.1 /usr/share/man/man8/chroot.8
sed -i s/\"1\"/\"8\"/1 /usr/share/man/man8/chroot.8
```

Деякі скрипти у пакеті LFS-Bootscripts залежать від **head, sleep** і **nice**. Так як /usr може не бути доступною при ранніх стадіях завантаження, ці бінарі повинні бути в кореневому розділі:

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

6.26.2. Вміст Coreutils

Встановлені програми: base64, basename, cat, chcon, chgrp, chmod, chown, chroot, cksum, comm, cp, csplit, cut, date, dd, df, dir, dircolors, dirname, du, echo, env, expand, expr, factor, false, fmt, fold, groups, head, hostid, id, install, join, link, ln, logname, ls, md5sum, mkdir, mkfifo, mknod, mktemp, mv, nice, nl, nohup, nproc, od, paste, pathchk, pinky, pr, printenv, printf, ptx, pwd, readlink, realpath, rm, rmdir, runcon, seq, sha1sum, sha224sum, sha256sum, sha384sum, sha512sum, shred, shuf, sleep, sort, split, stat, stdbuf, stty, sum, sync, tac, tail, tee, test, timeout, touch, tr, true, truncate, tsort, tty, uname, unexpand, uniq, unlink, users, vdir, wc, who, whoami і yes

Встановлені бібліотеки: libstdbuf.so

Встановлені директорії: /usr/libexec/coreutils

Короткий опис

base64 Закодовує чи декодує дані згідно з специфікацією base64 (RFC3548)

basename Видаляє шлях даний як суфікс з імені файлу

cat Об'єднує файли у стандартний вивід

chcon Змінює контекст безпеки для файлів і директорій

chgrp Змінює власництво групи файлів і каталогів

chmod Змінює права кожного файлу до заданого режиму; Режим може бути

	символічним представленням змін
chown	Змінює власника або/і групу файлів чи директорій
chroot	Виконує команду з заданою директорією як кореневою
cksum	Виводить перевірочну суму Cyclic Redundancy Check (CRC) і суму байтів для кожного заданого файлу
comm	Порівнює два відсортовані файли, виводить у три колонки рядки які є унікальними і рядки які є спільними
cp	Копіює файли
csplit	Розділяє заданий файл на декілька файлів, діляци їх у відповідності до даного шаблону чи номерів рядків і виводить рахунок байтів кожного нового файлу
cut	Виводить секції рядків, вибираючи частини у відповідності до полів чи позицій
date	Виводить поточний час у заданому форматі, чи встановлює системний час
dd	Копіює файл використовуючи заданий розмір блоків і рахунок, під час опціонального його перетворення
df	Виводить суму доступного дискового простору (і використаного) на усіх монтованих файлових системах, чи тільки на заданій файловій системі
dir	Виводить вміст заданої директорії (так само як команда **ls**)
dircolors	Виводить команди які встановлюють змінну середовища LS_COLOR для зміни схеми кольорів, які використовуються командою **ls**
dirname	Видаляє суфікс імен не директорій з назв файлів
du	Повідомляє суму дискового простору, який використовується поточною директорією, кожною даною директорією (включаючи усі підкаталоги) або кожним з даних файлів
echo	Виводить даний їй рядок
env	Виконує команду з модифікованими змінними середовища
expand	Перетворює табуляції у пробіли
expr	Виконує вираження
factor	Виводить прості множини для усіх цілих чисел

false	Не робить нічого, вона завжди існує з кодом виходу який вкащує помилку
fmt	Переформатовує параграфи у даному файлі
fold	Перетворює кожний рядок у даному файлі
groups	Повідомляє членство користувача у групах
head	Виводить перші десять рядів (чи іншу кількість) кожного даного файлу
hostid	Повідомляє чисельний ідентифікатор (у шістнадцятковій системі числення) хосту
id	Повідомляє ефективний ID користувача, ID групи і членство у групах поточного чи специфікованого користувача
install	Копіює файли під час встановлення їхніх прав доступу і, якщо можливо, їхнього власника і групу
join	Додає рядки які мають ідентичні поля з'єднання з двох окремих файлів
link	Створює жорстке посилання з заданим ім'ям до файлу
ln	Робить жорсткі або символічні посилання між файлами
logname	Повідомляє поточне ім'я користувача
ls	Перераховує вміст кожної даної директорії
md5sum	Повідомляє чи перевіряє суми Message Digest 5 (MD5)
mkdir	Створює директорії з заданим ім'ям
mkfifo	Створює файл FIFO (First-In, First-Out) з заданим ім'ям
mknod	Створює файл пристрою з заданим ім'ям; файл пристрою є спеціальний символьний файл, спеціальний блоковий файл чи FIFO
mktemp	Створює тимчасові файли у безпечній манері; вона використовується скриптами
mv	Переміщає чи перейменовує файли або директорії
nice	Виконує програму з модифікованими пріоритетами
nl	Зчитує кількість рядків з даного файлу
nohup	Виконує команду з імунітетом до зависань, з її перенаправленим виводом у файл журналу

nproc	Виводить число одиниць оброблювання які доступні для процесу
od	Вивантажує файл у вісімковій системі числення чи інших форматах
paste	Об'єднує дані файли, послідовно з'єднуючи рядок за рядком, розділеними табуляцією
pathchk	Перевіряє чи імена файлів є дійсними чи портативними
pinky	Легкий клієнт finger, він виводить деяку інформацію про даного користувача
pr	Розбиває файл на сторінки чи колонки для друку
printenv	Виводить змінні робочого середовища
printf	Виводить дані аргументи у відповідності до даного формату, більш як функція printf мови C
ptx	Продукує переставлений зміст з вмісту даного файлу, з кожним ключовим словом у його контексті
pwd	Повідомляє ім'я поточної робочої директорії
readlink	Повідомляє значення даного символічного посилання
realpath	Виводить витягнутий шлях
rm	Видаляє файли чи директорії
rmdir	Видаляє директорії, якщо вони є пустими
runcon	Виконує команду з заданим контекстом безпеки
seq	Виводить послідовність чисел даного рангу і з даним інкрементом
sha1sum	Виводить чи перевіряє 160-бітний перевірочну суму SHA1 (Secure Hash Algorithm)
sha224sum	Виводить чи перевіряє 224-бітний перевірочну суму SHA (Secure Hash Algorithm)
sha256sum	Виводить чи перевіряє 256-бітний перевірочну суму SHA (Secure Hash Algorithm)
sha384sum	Виводить чи перевіряє 384-бітний перевірочну суму SHA
sha512sum	Виводить чи перевіряє 512-бітний перевірочну суму SHA

shred	Перезаписує дані файли повторно з комплексними шаблонами, роблячи складним відновлення даних
shuf	Перемішує рядки тексту
sleep	Призупиняє процес на заданий час
sort	Сортує рядки з даного файлу
split	Розбиває даний файл на куски, за розміром чи за номером рядків
stat	Виводить статус файлової системи чи файлу
stdbuf	Виконує команди з заміненими буферними операціями для їх стандартних потоків
stty	Встановлює чи повідомляє параметри термінального рядка
sum	Виводить перевірочну суму і кількість блоків для кожного даного файлу
sync	Скидає буфери файлової системи; вона змушує запис змінених блоків на диск і оновлює супер блок
tac	З'єднує дані файли у зворотньому порядку
tail	Виводить останні десять рядків (чи дане число рядків) для кожного даного файлу
tee	Читає з стандартного вводу поки записує обидва до стандартного виводу і до даних файлів
test	Порівнює значення і перевіряє типи файлів
timeout	Виконує команду з лімітом часу
touch	Змінює часові мітки файлів, встановлюючи час модифікування і доступу даного файлу до поточного часу; файли, які не існують, створюються з нульовим розміром
tr	Перетворює, вичавлює і видаляє дані символи з стандартного вводу
true	Не робить нічого, завжди повертає код статусу завершення, який означає успіх
truncate	Стискує чи розширює файли до заданого розміру
tsort	Виконує топологічне сортування; вона записує повністю впорядкований список у файл

tty	Повідомляє ім'я файлу терміналу підключеного до стандартного вводу
uname	Повідомляє системну інформацію
unexpand	Перетворює пробіли у табуляції
uniq	Відкидає усі рядки крім успішно знайдених
unlink	Видаляє даний файл
users	Виводить імена користувачів які присутні у системі
vdir	Так само як і **ls -l**
wc	Повідомляє номера рядків, слів і байтів для кожного файлу, так само як і суму рядків даних файлів
who	Повідомляє хто є в системі
whoami	Повідомляє асоційоване ім'я користувача з поточним ефективним ID користувача
yes	Багаторазово виводить “y” чи даний рядок поки процес не буде вбитий
`libstdbuf.so`	Бібліотека, яка використовується **stdbuf**

6.27. Iana-Etc-2.30

Пакет Iana-Etc постачає дані для мережних сервісів і протоколів.

Приблизний час побудови: менш ніж 0.1 SBU
Необхідний дисковий простір: 2.2 Мбайт

6.27.1. Встановлення Iana-Etc

Наступні команди перетворюють “сирі” дані, які постачаються IANA у коректний формат для файлів даних `/etc/protocols` і `/etc/services`:

```
make
```

Цей пакет не постачається з тестовим набором

Встановіть пакет:

```
make install
```

6.27.2. Вміст Iana-Etc

Встановлені файли: `/etc/protocols` і `/etc/services`

Короткий опис

/etc/protocols	Описує деякі протоколи DARPA, які є доступними з підсистеми TCP/IP
/etc/services	Забезпечує карту між дружнім текстовим представленням інтернет сервісів і їхніми прикріпленими номерами портів і типом протоколів

6.28. M4-1.4.16

Пакет M4 вміщує макро процесор

Приблизний час побудови: 0.4 SBU
Необхідний дисковий простір: 26.6 Мбайт

6.28.1. Встановлення M4

Виправте несумісність між цим пакетом і Glibc-2.16.0:

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

Підготуйте M4 до компіляції:

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

Скомпілюйте пакет:

```
make
```

Щоб виконати тести, спочатку необхідно виправити тестову програму і тоді виконайте тести:

```
sed -i -e '41s/ENOENT/& || errno == EINVAL/' tests/test-readlink.h
make check
```

Встановіть пакет:

```
make install
```

6.28.2. Вміст M4

Встановлена програма: m4

Короткий опис

м4 копіює дані файли поки розширює макроси які вони мають у собі. Ці макроси або вбудовані, або визначені користувачем і можуть мати будь-яку кількість аргументів. Крім того, виконуючи макро розширення, **м4** має вбудовані функції для підключення файлів, виконання команд Unix, цілочисельної арифметики, маніпулювання текстом, рекурсією тощо. Програма **м4** може використовуватися або як інтерфейс до компілятора, або як макропроцесор.

6.29. Bison-2.6.2

Пакет Bison вміщує генератор синтаксичного аналізатора.

Приблизний час побудови: 1.3 SBU
Необхідний дисковий простір: 34 Мбайт

6.29.1. Встановлення Bison

Підготуйте Bison до компіляції:

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

Система конфігурації спричиняє побудову Bison без підтримки інтернаціоналізації повідомлень помилок якщо програма bison не є у $PATH. Наступна додаткова команда виправить це:

```
echo '#define YYENABLE_NLS 1' >> lib/config.h
```

Скомпілюйте пакет:

```
make
```

Для тестування результатів (приблизно 0.5 SBU), виконайте:

```
make check
```

Встановіть пакет:

```
make install
```

6.29.2. Вміст Bison

Встановлені програми: bison і yacc

Встановлені бібліотеки: liby.a

Встановлені директорії: /usr/share/bison

Короткий опис

bison Генерує, з серії правил, програму для аналізування структури текстових файлів; Bison є заміною для yacc (Yet Another Compiler Compiler)

yacc Обгортка для **bison**, існує для програм, які досі використовують **yacc** замість **bison**; вона викликає **bison** з параметром -y

`liby.a` Бібліотека Yacc, яка вміщує реалізацію сумісними з Yacc функціями yyerror і main; ця бібліотека за нормальних обставин не є дуже корисною, але POSIX вимагає її

6.30. Procps-3.2.8

Пакет procps вміє програми для моніторингування процесів

Приблизний час побудови: 0.1 SBU
Необхідний дисковий простір: 2.6 Мбайт

6.30.1. Встановлення Procps

Застосуйте патч для запобігання виведенняя повідомлень про помилки при визначенні швидкості годинника ядра:

```
patch -Np1 -i ../procps-3.2.8-watch_unicode-1.patch
```

Застосуйте патч щоб виправити проблему з підтримкою unicode у програму watch:

```
patch -Np1 -i ../procps-3.2.8-fix_HZ_errors-1.patch
```

Виправте помилки у файлі Makefile, який запобігає будуванню procps з make-3.82:

```
sed -i -e 's@\*/module.mk@proc/module.mk ps/module.mk@' Makefile
```

Скомпілюйте пакет:

```
make
```

Цей пакет не постачається з тестовим набором.

Встановіть пакет:

```
make install
```

6.30.2. Вміст Procps

Встановлені програми: free, kill, pgrep, pkill, pmap, ps, pwdx, skill, slabtop, snice, sysctl, tload, top, uptime, vmstat, w, and watch

Встанволені бібліотеки: `libproc.so`

Короткий опис

free Повідомляє кількість вільбної і використаної пам'яті (фізичної і області підкачки) у системі

kill Посилає сигнали до процесів

pgrep Шукає процеси за їхніми іменами і іншими атрибутами

pkill Посилає сигнали процесам за їхніми іменами і іншими атрибутами

pmap Виводить карту пам'яті для даного процесу

ps Перераховує поточні виконувані процеси

pwdx Повідомляє поточну робочу директорію процесу

skill	Посилає сигнали до процесів, які відповідають певним критеріям
slabtop	Виводить детальну інформацію проб кешу ядра у реальному часі
snice	Змінює пріоритет процесу, який відповідає заданому критерію
sysctl	Модифікує параметри ядра під час виконання
tload	Виводить графік поточного завантаження системи
top	Виводить список найбільш інтенсивних процесів, вона забезпечує постійний показ виконання процесів у реальному часі
uptime	Повідомляє кількість часу роботи системи, кількість користувачів присутніх у системі і завантаження системи
vmstat	Повідомляє статистику використання віртуальної пам'яті, даючи інформацію про процеси, пам'ять, сторінки, блоки вводу/виводу (IO) і активність процесора
w	Показує які користувачі присутні у системі, де і від коли
watch	Повторно виконує дану команду, показуючи її перший повноекранний вивід; це дозволяє користувачу переглядати зміну виводу крізь час
`libproc`	Вміщує функції, які використовуються більшістю програм у цьому пакеті

6.31. Grep-2.14

Пакет Grep вміщує програми для пошуку крізь файли

Приблизний час побудови: 0.4 SBU
Необхідний дисковий простір: 30 Мбайт

6.31.1. Встановлення Grep

Підготуйте Grep до встановлення

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

Скомпілюйте пакет

```
make
```

Для тестування результатів, виконайте:

```
make check
```

Встановіть пакет:

```
make install
```

6.31.2. Вміст Grep

Встановлені програми: egrep,fgrep і grep

Короткий опис

egrep Виводить рядки, які збігаються з розширеним регулярним виразом

fgrep Друкує рядки, які збігаються з списком фіксованих рядків

grep Виводить рядки, які збігаються з базовим регулярним виразом

6.32. Readline-6.2

Пакет Readline є набором бібліотек, які пропонують редагування командного рядка і можливості історії.

Приблизний час побудови: 0.1 SBU
Необхідний дисковий простір: 17.2 Мбайт

6.32.1. Встановлення Readline

Перевстановлення пакету Readline спричинить переміщення старих бібліотек до каталогу `<назва_бібліотеки>.old`. За нормальних умов це не є проблемою, але в деяких випадках вона можу спричинити помилку компонування у ldconfig. Це попереджається виконанням наступних двох команд:

```
sed -i '/MV.*old/d' Makefile.in
sed -i '/{OLDSUFF}/c:' support/shlib-install
```

Застосуйте патч для виправлення відомої помилки, яка була виправлена потоком:

```
patch -Np1 -i ../readline-6.2-fixes-1.patch
```

Підготуйте Readline до компіляції:

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

Скомпілюйте пакет:

```
make SHLIB_LIBS=-lncurses
```

Значення опцій make:

SHLIB_LIBS=-lncurses

Ця опція змушує Readline компонуватися з бібліотекою libncurses (насправді з libncursesw)

Цей пакет не постачається з тестовим набором.

Встановіть пакет:

```
make install
```

Тепер перемістіть статичні бібліотеки для пільш правильного розміщення:

```
mv -v /lib/lib{readline,history}.a /usr/lib
```

Наступним чином, видаліть файли .so у /lib і змініть їхні посилання їх у /usr/lib

```
rm -v /lib/lib{readline,history}.so
ln -sfv ../../lib/libreadline.so.6 /usr/lib/libreadline.so
ln -sfv ../../lib/libhistory.so.6 /usr/lib/libhistory.so
```

За бажанням, встановіть документацію:

```
mkdir -v /usr/share/doc/readline-6.2
install -v -m644 doc/*.{ps,pdf,html,dvi} \
/usr/share/doc/readline-6.2
```

6.32.2. Вміст Readline

Встановлені бібліотеки: libhistory.{a,so} і libreadline.{a,so}

Встановлені директорії: /usr/include/readline, /usr/share/readline, /usr/share/doc/readline-6.2

Короткий опис

`libhistory`	Постачає сумісний інтерфейс користувача для виклику рядків історії
`libreadline`	Допомагає у узгодженості інтерфейсу користувача між дискретними програмами, які мають забезпечувати інтерфейс командного рядка

6.33. Bash

Пакет Bash вміщує Bourne-Again Shell

Приблизний час побудови: 1.7 SBU
Необхідний дисковий простір: 45 Мбайт

6.33.1. Встановлення Bash

Для початку, застосуйте наступні патчі для виправлення деяких помилок, які були вказані потоком:

```
patch -Np1 -i ../bash-4.2-fixes-8.patch
```

Підготуйте Bash до компіляції

```
./configure —prefix=/usr \
--bindir=/bin \
--htmldir=/usr/share/doc/bash-4.2 \
--without-bash-malloc \
--with-installed-readline
```

Значення опцій конфігурування

-htmldir

Ця опція позначає директорію у яку форматована у HTML документація буде встановленою

--with-installed-readline

Ця опція вказує bash використ овувати бібліотеку readline, яка вжу встановлена у системі замість використання її власної версії readline

Скомпілюйте пакет:

```
make
```

Пропустіть встановлення пакету, якщо ви не виконуєте тестовий набір програм

Для підготовки тестів, впевніться, що користувач nobody може робити запис у дерево джерела коду:

```
chown -Rv nobody
```

Тепер, виконайте тести як користувач nobody:

```
su nobody -s /bin/bash -c "PATH=$PATH make tests"
```

Встановіть пакет:

```
make install
```

Виконайте нову програму bash (заміняючи ту, яка зараз виконується):

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

Увага

Параметри які використовувалися для bash, щоб зробити його процесом інтерактивної оболонки входу і продовження вимкнення хешування для того, щоб нові програми були знайденими коли вони стають доступними

6.33.2. Вміст Bash

Встановлені програми: bash, bashbug і sh (посилання на bash)

Встановлені програми: `/usr/share/doc/bash-4.2`

Короткий опис

bash Широко використовуваний командний інтерпретатор, він виконує широкий спектр розбирань і замін на даному командному рядку перед тим, як виконати його, це робить інтерпретатор могутнім інструментом

bashbug Скрипт командного рядка, яка допомагає користувачу складати і відсилати по електронній пошті стандартно форматовані звістки про помилки **bash**

sh Посилання до програми **bash**; коли викликається програма **sh**, **bash** намагається копіювати поведінку запуску історичних версій **sh** так близько, як це можливо, у відповідності до стандарту POSIX

6.34. Libtool-2.4.2

Пакет Libtool вміщує загальну бібліотеку підтримки скриптів GNU. Вона обгортає комплексність використання сумісних динамічних бібліотек, переносний інтерфейс.

Приблизний час побудови: 3.0 SBU
Необхідний дисковий простір: 37 Мбайт

6.34.1. Встановлення Libtool

Підготуйте Libtool до встановлення:

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

Скомпілюйте пакет

```
make
```

Для тестування результатів (приблизно 3.0 SBU), виконайте:

```
make check
```

Встановіть пакет:

```
make install
```

6.34.2. Вміст Libtool

Встановлені програми: libtool і libtoolize

Встановлені бібліотеки: `libltdl.{a,so}`

Встановлені директорії: `/usr/include/libltdl, /usr/share/libtool`

Короткий опис

libtool Постачає генеровану підтримку бібліотечно побудованих сервісів

libtoolize Забезпечує стандартний шлях для додавання підтримки **libtool** до пакету

`libltdl` Приховує деякі складнощі розробки динамічних бібліотек

6.35. GDBM-1.10

Пакет GDBM вміщує GNU Database Manager. Це є форматована дискова база даних, яка зберігає ключі/дані-пари у одному файлі. Фактичні дані будь-якого запису зберігаються є індексом з унікальним ключом, який може бути отриманим за менший час, ніж він би був отриманий у текстовому файлі.

Приблизний час побудови: 0.1 SBU
Необхідний дисковий простір: 8.5 Мбайт

6.35.1. Встановлення GDBM

Підготуйте GDBM до компіляції

```
./configure --prefix=/usr --enable-libgdbm-compat
```

Значення опцій конфігурування:

`--enable-libgdbm-compat`

Цей прапореця вмикає сумісну бібліотеку libgdbm до побудови, так як деякі пакети ззовні LFS можуть вимагати старіші підпрограми DBM

Скомпілюйте пакет:

```
make
```

Для тестування результатів, виконайте:

```
make check
```

Встановіть пакет:

```
make install
```

6.35.2. Вміст GDBM

Встановлені програми: testgdbm

Встановлені бібліотеки: `libgdbm.{so,a} i libgdbm_compat.{so,a}`

Короткий опис

testgdbm Тестує і модифікує базу даних GDBM

libgdbm Вміщує функції для маніпулювання хешованою базою даних

6.36. Inetutils-1.9.1

Пакет Inetutils вміщує програми для базової підтримки мережі.

Приблизний час побудови: 0.4 SBU
Необхідний дисковий простір: 27 Мбайт

6.36.1. Встановлення Inetutils

Виправте несумісність між цим пакетом і Glibc-2.16.0

Підготуйте Inetutils до компіляції:

```
./configure --prefix=/usr \
--libexecdir=/usr/sbin \
--localstatedir=/var \
--disable-ifconfig \
--disable-logger \
--disable-syslogd \
--disable-whois \
--disable-servers \
```

Значення опцій конфігурування:

`--disable-ifconfig`

Ця опція запобігає Inetutils від встановлення програми ifconfig, яка може бути використаною для конфігурації мережних інтерфейсів. LFS використовує ip з IPRoute2 для виконання цих завдань.

`--disable-logger`

Ця опція зпапобігає Inetutils від встановлення програми logger, яка використовується скриптами для подання повідомлень до системного демона журналів. Не встановлюйте його тому, що Util-linux встановила їхню версію раніше.

`--disable-syslogd`

Ця опція запобігає Inetutils від встановлення системного демона журналу, який встановлений з пакетом Sysklogd.

`--disable-whois`

Ця опція вимикає побудову клієнта Inetutils whois, який є застарілим. Інструкції для кращого клієнта whois є у книзі BLFS.

`--disable-servers`

Це вимикає встановлення деяких мережних серверів, які включені як частина пакету Inetutils. Ці сервери вважаються невідповідними до базової системи LFS. Деякі з них є небезпесними за своєю природою і вважаються безпечними на довірених мережах. Докладніша інформація може бути знайдена ща адресою http://www.linuxfromscratch.org/blfs/view/svn/basicnet/inetutils.html. Зверніть увагу на те, що кращі заміни є доступними для більшості цих серверів.

Скомпілюйте пакет:

```
make
```

Для тестування результатів, виконайте:

```
make check
```

Встановіть пакет:

```
make install
make -C doc html
make -C doc install-html docdir=/usr/share/doc/inetutils-1.9.1
```

Перемістіть декі програми до їхніх сумісних з FHS місць:

```
mv -v /usr/bin/{hostname,ping,ping6} /bin
mv -v /usr/bin/traceroute /sbin
```

6.36.2. Вміст Inetutils

ftp	Програма протоколу передачі
hostname	Повідомляє чи встановлює ім'я хосту
ping	Посилає пакети ехо-запитів і повідомляє скільки часу зайняла відповідь
ping6	Версія ping для мереж IPv6
rcp	Виконує віддалене копіювання файлів
rexec	Виконує команди на віддаленому хості
rlogin	Виконує віддалений вхід в систему
rsh	Запускає віддалений командний інтерпретатор
talk	Використовується для спілкування з іншим користувачем
telnet	Інтерфейс до протоколу TELNET
tftp	Тривіальна програма передачі
traceroute	Трасує маршрут пакетів взятих з хосту на якому ви працюєте до іншого хосту мережі, показуючи усі проміжні хости (шлюзи) на шляху

6.37 Perl-5.16.1

Пакет Perl вміщує мову Parl Extraction and Report Language

Приблизний час побудови: 7.2 SBU
Необхідний дисковий простір: 247 Мбайт

6.37.1. Встановлення perl

Для початку створіть базовий файл /etc/hosts на який буде посилатися один з конфігураційних файлів Perl:

```
echo "127.0.0.1 localhost $(hostname)" > /etc/hosts
```

Ця версія Perl тепер будує модуль Compress::Raw::Zlib. За замовчуванням, Perl буде використовувати внутрішню копію коду Zlib для побудови. Виконайте наступну команду, щоб Perl використовував бібліотеку Zlib яка в становлена на системі:

```
sed -i -e "s|BUILD_ZLIB\s*= True|BUILD_ZLIB = False|" \
-e "s|INCLUDE\s*= ./zlib-src|INCLUDE = /usr/include|" \
-e "s|LIB\s*= ./zlib-src|LIB = /usr/lib|" \
cpan/Compress-Raw-Zlib/config.in
```

Для отримання повного контролю над процесом встановлення Perl, ви можете видалити "-des" опцію з наступної команди і власноручно вибрати певні опції побудови. Як альтернатива, використайте команду точно так само, як показано нижче для використання параметрів, які Perl автоматично визначить:

```
sh Configure -des -Dprefix=/usr
-Dvendorprefix=/usr
-Dman1dir=/usr/share/man/man1
-Dman3dir=/usr/share/man/man3
-Dpager="/usr/bin/less -isR"
-Duseshrplib
```

Значення опцій конфігурування:

-Dvendorprefix=/usr

Це запевняє, що перл знає куди вказувати пакетам Perl встановлювати їхні модулі.

-Dpager="/usr/bin/less -isR"

Це корегує помилку у шляху в який perldoc викликає програму less/

-Dman1dir=/usr/share/man/man1 -Dman3dir=/usr/share/man/man3

Так як Groff ще не встановлений, Configure визначить, що ми не хочемо встановлювати сторінки допомоги для Perl. Використовуючи ці параметри ми вирішуємо це питання.

-Duseshrplib

Побудувати розділювану бібліотеку libperl, яка необхідна для деяких модулів perl

Скомпілюйте пакет

```
make
```

Для тестування результатів (приблизно 2.5 SBU), виконайте:

```
make -k test
```

Встановіть пакет:

```
make install
```

6.37.2 Вміст Perl

Встановлені програми: a2p, c2ph, config_data, corelist, cpan, cpan2dist, cpanp, cpanp-run-perl, dprofpp, enc2xs, find2perl, h2ph, h2xs, instmodsh, json_pp, libnetcfg, perl, perl5.16.1 (link to perl), perlbug, perldoc, perlivp, perlthanks (link to perlbug), piconv, pl2pm, pod2html, pod2latex, pod2man, pod2text, pod2usage, podchecker, podselect, prove, psed (link to s2p), pstruct (link to c2ph), ptar, ptardiff, ptargrep, s2p, shasum, splain і xsubpp

Встановлені бібліотеки: Кілька сотень, що не може бути описаним тут

Встановлена директорія: /usr/lib/perl5

Короткий опис

a2p Перекладає awk до Perl

c2ph Робить дамп структур C згенерованих з cc -g -S

config_data Показує чи змінює конфігурацію модулів perl

corelist Командний інтерфейс для Module::CoreList

cpan Взаємодіє з Comprehensive Perl Archive Network (CPAN) з командного рядка

cpan2dist Конструктор дистрибутива CPANPLUS

cpanp програма, яка запускає CPANPLUS

cpanp-run-perl Скрипт Perl який використовується для увімкнення скидання буферу виводу після кожного запису у кожному породженому процесі

dprofpp Виводить профільні дані Perl

enc2xs Будує розширення Perl для модуля Encode з Unicode Character Mappings чи Tcl Encoding Files

h2ph Перетворює заголовкові файли C .h у заголовкові файли Perl .ph

h2xs Перетворює заголовкові файли C .h у розширення Perl

instmodsh Скрипт інтерпретатора для перевірки встановлених модулів Perl, який може створювати архів з встановлених модулів

json_pp Перетворює дані з певних фіорматів вводу-виводу

libnetcfg Можу бути використаним для конфігурації модуля Perl libnet

perl	комбінує декі найкращі особливості C, sed, awk чи sh у одну мову
perl5.16.1	Жорстке посилання до perl
perlbug	Використовується для генерування повідомлення про помилку Perl, чи модулів які постачаються з ним, і після цього відсилає їх по пошті
perldoc	Виводить частину документації, які є вбудованими у дерево встановлення Perl чи у скрипті Perl
perlivp	Perl Installation Verification Procedure (процедура перевірки встановлення Perl); вона може використовуватися для перевірки того, що Perl і його бібліотеки встановлені коректно
perlthanks	Використовується для генерації повідомлень подяки для відсилання до розробників Perl
piconv	Версія Perl перетворювача коду символів iconv
pl2pm	Груба утиліта для перетворення файлів Perl4 .pl у модулі Perl5 .pm
pod2html	Перетворює файли з формату pod у формат HTML
pod2latex	Перетворює файли з формату pod у формат LaTeX
pod2man	Перетворює файли з формату pod у форматований ввід *roof
pod2text	Перетворює дані формату pod у форматований текст ASCII
pod2usage	Виводить повідомлення з вбудованих документів pod у файли
podchecker	Перевіряє синтаксис для файлів документації формату pod
podselect	Виводить вибрані секції pod документації
prove	Утиліта командного рядка для виконання тестів модуля Test::Harmess
psed	Версія Perl редактора потоку sed
ptar	tar-подібна прошрама написана на perl
ptardiff	Програма Perl, яка порівнює розпакований архів з нерозпакованим
ptargrep	Програма Perl, яка застосовує збіг з шаблоном контент файлів у архіві
s2p	Перетворює sed скрипти у Perl
shasum	Виводить чи перевіряє перевірочні суми SHA

spain	Використовується для змушування розширеного діагностування повідомлень у Perl
xsupbb	Перетворює код Perl XS у код мови C

6.38 Autoconf-2.69

Пакет Autoconf вміщує програми для створення скриптів командного інтерпретатора, які можуть автоматично сконфігурувати дерево коду

Приблизний час побудови: 4.5 SBU
Необхідний дисковий простір: 17.1 Мбайт

6.38.1 Встановлення Autoconf

Підготуйте Autoconf до встановлення:

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

Скомпілюйте пакет

```
make
```

Для тестування результатів, виконайте:

```
make check
```

Цей займе багато часу — приблизно 4.7 SBU. У добавок, 6 тестів, які використовує Autoconf, є пропущеними. Для повного тестування, Autoconf може піддатися тестам після встановлення Automake.

Встановіть пакет:

```
make install
```

6.38.2 Вміст Autoconf

Встановлені програми: autoconf, autoheader, autom4te, autoreconf, autoscan, autoupdate, and ifnames

Встановені директорії: /usr/share/autoconf

Короткий опис

autoconf Створює скрипти оболонки, які автоматично конфігурують вихідний код пакетів програмного забезпечення для адаптування до великої кількості Юнікс-подібних систем. Конфігураційні скрипти, які він створює є незалежними їхнє виконання не вимагає програми autoconf

autoheader Утиліта для створення шаблонних файлів для тверджень мови C #define для використання як конфігураційний механізм

autom4te Обгортка для макропроцесора M4

autoreconf Автоматично виконує autoconf, aoutoheader, aclocal, automake, gettextize і libtoolize у правильному порядку для збереження часу при внесенні змін у файли шаблонів autoconf і automake

autoscan Допомагає створювати файл configure.in для пакетів програмного забезпечення; вона перевіряє файли коду у дереві каталогів, шукаючи у них проблеми з портуванням і створює файл configure.scan який обслуговується як попередній файл configure.in для пакету

autoupdate Модифікує файли configure.in які досі викликають макрос autoconf за їхніми старими назвами

ifnames Програма допомагає написати configure.in для пакету програмного забезпечення; вона виводить ідентифікатори, які використовує пакет за умовах препроцесора мови C. Якщо пакет налаштований для певної переносності, ця програма допомогає визначити який скрипт configure потребує перевірки. Також він допомогає заповнити прогалини у файлі configure.in згенерованим autoscan

6.39. Automake-1.12.3

Пакет Automake вміщує програми для генерації файлів Makefile для використання програмою Autocnv.

Приблизний час побудови: менш ніж 0.1 SBU (34.1 SBU з тестами)
Необхідний дисковий простір: 100 Мбайт

6.39.1 Встановлення Automake

Підготуйте Automake до компіляції:

```
./configure –prefix=/usr –docdir=/usr/share/doc/automake-1.12.3
```

Скомпілюйте пакет:

```
make
```

> Увага
>
> Виконання тестового пакету займе багато часу: більш ніж 30 SBU. Виконання тесту не рекомендовано.

Для тестування результатів побудови, виконайте:

```
make check
```

Встановіть пакет:

```
make install
```

6.39.2 Вміст Automake

Встановлені програми: acinstall, aclocal, aclocal-1.12, automake, automake-1.12, compile, config.guess, config.sub, depcomp, elisp-comp, install-sh, mdate-sh, missing, mkinstalldirs, py-compile, symlink-tree, і ylwrap

Встановлені директорії: /usr/share/aclocal-1.12, /usr/share/automake-1.12, /usr/share/doc/automake-1.12.3

Короткий опис

acinstall Скрипт який встановлює файли M4 у стилі aclocal

aclocal Генерує файли aclocal.m4 засновані на вмісті файлів configure.in

aclocal-1.12 Жорстке посилання на aclocal

automake Уиліта, яка автоматично генерує файли Makefile.in з Makefile.am. Для створення усіх файлів Makefile.in для пакету, виконайте цю програму у кореневій директорії дерева коду. Скануючи файл configure.in, вона автоматично знаходить кожен відповідний файл Makefile.am і генерує відповідний файл Makefile.in

automake-1.12 Жорстке посилання на automake

compile	Обгортка для компіляторів
config.guess	Скрипт який намагається вгадати канонічну трійню для даної побудови, хосту чи цільову архітектуру
config.sub	Конфігураційний ратифіковуючий підпрограмний скрипт
depcomp	Скрипт для компіляції програми, його інформація залежностей згенерована у додаток до згенерованого виводу
elisp-comp	Байт-компілює код Emacs List
install-sh	Скрипт, який встановлює програму, скрипт чи файл даних
mdate-sh	Скрипт, який виводить час модифікації файлу чи директорії
missing	Скрипт, який веде себе як загальну заглушку для відсутніх програм GNU під час встановлення
mkinstalldirs	Скрипт, який створює дерево каталогів
py-compile	Компілює програми Python
symlink-tree	Скрипт який створює дерево посилань на дерево каталогів
ylwrap	Обгортка для lex і yacc

6.40 Diffutils-3.2

Пакет Diffutils вміщує програми, які показують різницю між файлами і директоріями

Приблизний час побудови: 0.5 SBU
Необхідний дисковий простір: 25 Мбайт

6.40.1 Встановлення Diffutils

Виправте несумісність між цим пакетом і Glibc-2.16.0

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

Підготуйте Diffutils до компіляції:

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

Скомпілюйте пакет:

```
make
```

Для тестування результатів, виконайте:

```
make check
```

Встановіть пакет

```
make install
```

6.40.2 Вміст Diffutils

Встановлені програми: cmp, diff,diff3 і sdiff

Короткий опис

cmp Порівнює два файли і повідомляє де чи на яких байтах вони відрізняються

diff Порівнює два файли чи каталоги і повідомляє які лінії у файлах різні

sdiff Поєднує два файли і інтерактивно виводить результати

6.41 Gawk-4.0.1

Пакет Gawk вміщує програми для маніпулювання текстовими файлами.

Приблизний час побудови: 0.2 SBU
Необхідний дисковий простір: 30 Мбайт

6.41.1 Встановлення Gawk

Підготуйте Gawk до компілювання:

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

Скомпілюйте пакет:

```
make
```

Для тестування результатів, виконайте:

```
make check
```

Встановіть пакет:

```
make install
```

За бажанням, встановіть документацію

```
mkdir -v /usr/share/doc/gawk-4.0.1
cp -v doc/{awkforai.txt,*.{eps,pdf,jpg}} /usr/share/doc/gawk-4.0.1
```

6.41.2 Вміст Gawk

Встановлені програми: awk (link to gawk), dgawk, gawk, gawk-4.0.1, grcat, igawk, pgawk, pgawk-4.0.1, і pwcat

Встановлені директорії: /usr/lib/awk, /usr/share/awk

Короткий опис

awk Посилання на gawk

dgawk Відлагоджувач awk

gawk Програма для маніпулювання текстовими файлами; реалізація GNU awk

gawk-4.0.1 Жорстке посилання на gawk

grcat Робть дамп бази даних груп файлу /etc/group

igawk Дає gawk можливість під'єднувати файли

pgawk Профільована версія gawk

pgawk-4.0.1 Жорстке посилання на pgawk

pwcat Робить дамп бази паролів `/etc/passwd`

6.42 Findutils-4.4.2

Пакет Findutils вміщує програми для знаходження файлів. Ці програми забезпечуютьь рекурсивний пошук крізь дерево директорій і для створення, підтримки і пошуку баз даних (які часто швидші ніж рекурсивний пошук, але не надійні, якщо бази даних часто не оновлюється).

Приблизний час побудови: 0.4 SBU
Необхідний дисковий простір: 29 Мбайт

6.42.1 Встановлення Findutils

Підготуйте Findutils до компіляції:

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

Значення опцій конфігурування:

--localstatedir

Ця опція змінює місце бази даних locate на /vat/lib/locate, що вимагається FHS

Скомпілюйте пакет:

```
make
```

Для тестування результатів, виконайте:

```
make check
```

Встановіть пакет:

```
make install
```

Деякі скрипти у пакеті LFS-Bootscripts залежать від find. Так як /usr може бути не доступним під час ранніх стадій завантаження системи, ця програма повинна бути розміщеною на кореневому розділі. Скрипт updatedb також має бути модифікованим для корегування явного шляху:

```
mv -v /usr/bin/find /bin
sed -i 's/find:=${BINDIR}/find:=\/bin/' /usr/bin/updatedb
```

6.42.2 Вміст Findutils

Встановлені програми: bigram, code, find, frcode, locate, oldfind, updatedb і xargs

Встановлені директорії: /usr/lib/findutils

Короткий опис

bigram Раніше використовувалося для ствренння баз даних locate

code Раніше використовувалося для створення баз даних locate; це є предок frcode

find Шукає у даному дереві каталогів файли, які відповідають зазначеним критеріям

frcode	Викликається updatedb для стиснення списку імен файлів; вона використовує перед стиснення, знижуючи розмір бази даних з фактором від чотирьох до п'яти
locate	Шукає крізь базу даних імен файлів і повідомляє імена, які вміщують даний рядок чи збіг за даним шаблоном
oldfind	Старіша версія find, яка використовує інший алгоритм
updatedb	Оновлює базу даних locate; вона сканує усю файлову систему (включаючи інші файлові системи, які монтовані в поточний час, окрім, якщо сказано це не робити) і вкладає кожне ім'я фіайлу у базу даних
xargs	Може використовуватися для застосування даної команди до списку файлів

6.43 Flex-2.5.37

Пакет Flex вміщує утиліту для генерування програм, які розпізнають шаблони в тексті.

Приблизний час побудови: 0.4 SBU
Необхідний дисковий простір: 39 Мбайт

6.43.1 Встановлення Flex

Для початку, виправте деякі регресійні тести:

```
patch -Np1 -i ../flex-2.5.37-bison-2.6.1-1.patch
```

Підготуйте Flex до компіляції

```
./configure --prefix=/usr --mandir=/usr/share/man --infodur=/usr/share/info
```

Скомпілюйте пакет:

```
make
```

Для тестування результатів (приблизно 0.5 SBU), виконайте:

```
make -k check
```

Є два тести які проваляться відповідно до несумісності з більшістю новими версіями bison.

Встановіть пакет:

```
make install
```

Існують деякі пакети, які очікують знайти бібліотеку lex у каталозі `/usr/lib`. Створіть посилання для вирішення цього:

```
ln -sv libfl.a /usr/lib/libl.a
```

Декілька програм не знають поки про flex і намагатимуться виконати його попередник, lex. Для підтримки цих програм, створіть скрипт-обгортку названу lex, яка викликає flex у режимі емуляції lex:

```
cat > /usr/bin/lex << "EOF"
#!/bin/sh
# Begin /usr/bin/lex
exec /usr/bin/flex -l "$@"
# End /usr/bin/lex
EOF
chmod -v 755 /usr/bin/lex
```

За бажанням, встановіть файл документації flex.pdf:

```
mkdir -v /usr/share/doc/flex-2.5.37
cp -v doc/flex.pdf /usr/share/doc/flex-2.5.37
```

6.43.2 Вміст Flex

Встановлені програми: flex і lex

Встановлені бібліотеки: libfl.a і libfl_pic.a

Короткий опис

flex	Утиліта для генерування програм, які розпізнають шаблони у тексті; вона дозволяє гнучко вказувати правила для знаходження шаблонів, використовуючи потребу для розробки спеціалізованих програмам
lex	Скрипт, який виконує flex у режимі емуляції lex
`libfl.a`	Бібліотека flex

6.44 Gettext-0.18.1.1

Пакет Gettext вміщує утиліти для інтернаціоналізації і локалізацій. Це дозволяє програмам бути компільованими з NLS (Native Language Suport — підтримка рідної мови), які дозволяють їм виводити повідомлення на рідній мові користувача.

Приблизний час побудови: 2.3 SBU
Необхідний дисковий простір: 180 Мбайт

6.44.1 Встановлення Gettext

Виправте несумісність між цим пакетом і Glibc-2.16.0:

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

Підготуйте Gettext до компіляції:

```
./configure --prefix=/usr \
--docdir=/usr/share/doc/gettext-0.18.1.1
```

Скомпілюйте пакет:

```
make
```

Для тестування результатів (цей займе багато часу, приблизно 3 SBU), виконайте:

```
make check
```

Встановлення пакету:

```
make install
```

6.44.2 Вміст Gettext

Встановлення програм: autopoint, config.charset, config.rpath, envsubst, gettext, gettext.sh, gettextize, hostname, msgattrib, msgcat, msgcmp, msgcomm, msgconv, msgen, msgexec, msgfilter, msgfmt, msggrep, msginit, msgmerge, msgunfmt, msguniq, ngettext, recode-sr-latin, і xgettext

Встановлені бібліотеки: libasprintf.{a,so}, libgettextlib.so, libgettextpo.{a,so}, libgettextsrc.so, і preloadable_libintl.so

Встановлені директорії: /usr/lib/gettext, /usr/share/doc/gettext-0.18.1.1, /usr/share/gettext

Короткий опис

autopoint Копіює стандартну інфраструктуру файлів у пакет джерела коду

config.charset Виводить системно-залежні таблиці за псевдонімами кодів символів

config.rpath Виводить системно-залежний набір змінних, який описує як встановити пошук шляхів динамічних бібліотек у виконуваному файлі

envsubst Замінює змінні середовища у формат рядків оболонки

gettext Перекладає натуральну мову повідомлень на мову користувача пошуком

	перекладів у каталозі повідомлень
gettext.sh	Головним чином обслуговує бібліотеку фінкцій оболонки для gettext
gettextize	Копіює усі стандартні файли Gettext у даний кореневий каталог пакету для початку його інтернаціоналізування
hostname	Виводить мережне ім'я у різних формах
msgattrib	Фільтрує повідомлення каталогу перекладів у відповідності до їхніх атрибутів і маніпулює атрибути
msgcat	З'єднує і зливає дані файли .po
msgcmp	Порівнює два файли .po для перевірки того, що обидва містять такі ж набори рядків msgid
msgcomm	Знаходить повідомлення, які є загальними дл файлів .po
msgconv	Перетворює каталог перекладів до даного кодування символів
msgen	Створює каталог перекладів Англійської мови
msgexec	Застосовує команду до усіх перекладів каталогу перекладів
msgfilter	Застосовує фільтер до усіх перекладів каталогу перекладів
msgfmt	Генерує бінарний каталог повідомлень з каталогу перекладів
msggrep	Витягує усі переклади каталогу перетворень, які співпадають з даним шаблоном чи мають відношення до даних файлів джерел коду
msginit	Створює новий файл .po, ініціалізуючи мета-інформацію з значенням з середовища користувача
msgmerge	Комбінує два необроблених переклади у один файл
msgunfmt	Декомпілює каталог бінарних повідомлень у необроблений переклад тексту
msguniq	Уніфікує дубльовані переклади у директорії перекладів
ngettext	Виводить рідні переклади текстових повідомлень, граматична форма яких залежить від чисел
recode-sr-latin	Перекодовує текст мови Сербії з кирилиці до латиниці
xgettext	Витягує перекладені повідомлення рядків з даного файлу джерела коду, щоб зробити перший шаблон перекладу

`libasprintf`	Визначає клас autosprintf, який робить форматований вивід підпрограм мови C доступними у програмах C++, для використання з рядками <string> і потоками <iostreams>
`libgettextlib`	Приватна бібліотека, яка містить загальні підпрограми, які використовуються деякими програмами Gettext; вони не призначені для загального користування
`libgettextpo`	Використовується для запису спеціалізованих програм які обробляють файли .po; ця бібліотека використовується коли стандартних програм з'єднананих з Gettext (такі як msgcomm, msgcmp, msgattrib і msgen) не достатньо.
`libgettextsrc`	Приватна бібліотека, яка вміщує загальні підпрограми, які використовуються деякими програмами Gettext; вони не призначені для загального користування
`preloadable_libntl`	Бібліотека, призначення для використання LD_PRELOAD, яка допомагає libintl

6.45 Groff-1.21

Пакет Groff вміщує програми для обробки і форматування тексту.

Приблизний час побудови: 0.5 SBU
Необхідний дисковий простір: 83 Мбайт

6.45.1 Встановлення Groff

Groff очікує, щоб змінна середовища PAGE містила стандартний розмір паперу. Для користувачів у Сполучених Штатах, PAGE=letter є відповідним. Інакше, PAGE=A4 може бути більш правильним. Поки стандартний розмір паперу сконфігурований під час компіляції, вона може бути переписаною пізніше виводом рядка “A4” чи “letter” у файл /etc/papersize.

Підготуйте Groff до компіляції:

```
PAGE=<розмір_паперу> ./configure --prefix=/usr
```

Скомпілюйте пакет

```
make
```

Цей пакет не постачається з тестовим набором.

Встановіть пакет:

```
make install
```

Деякі програми документації, такі як xman, не будуть працювати правильно без наступних посилань:

```
ln -sv eqn /usr/bin/geqn
ln -sv tbl /usr/bin/gtbl
```

6.45.2 Вміст Groff

Встановлені програми: addftinfo, afmtodit, chem, eqn, eqn2graph, gdiffmk, geqn (link to eqn), grap2graph, grn, grodvi, groff, groffer, grog, grolbp, grolj4, grops, grotty, gtbl (link to tbl), hpftodit, indxbib, lkbib, lookbib, mmroff, neqn, nroff, pdfroff, pfbtops, pic, pic2graph, post-grohtml, preconv, pre-grohtml, refer, roff2dvi, roff2html, roff2pdf, roff2ps, roff2text, roff2x, soelim, tbl, tfmtodit, and troff

Встановлені директорії: /usr/lib/groff, /usr/share/doc/groff-1.21, /usr/share/groff

Короткий опис

addftinfo Читає файл шрифту troff і додає деяку додаткову інформацію, яка використовується системою groff

afmtodit Створює файл шрифту для використання groff і grops

chem Препроцесор Groff для створення діаграм хімічних структур

eqn Компілює дескриптори рівнянь вбудованих в файли вводу troff у команди, які є зрозумілими для troff

eqn2graph	Перетворює troff EQN (equation) у обрізане зображення
gdiffmk	Робить мітки різниці між файлами groff/nroff/troff
geqn	Посилання на eqn
grap2graph	Перетворює діаграми grap у обрізане зображення
grn	Препроцесор groff для файлів gremlin
grodvi	Драйвер для groff який створює формат Tex dvi
groff	Інтерфейс до системи форматування документів groff; зазвичай, вона запускає програму troff і постпроцесор відповідний для вибраного пристрою
groffer	Виводить файли groff і сторінки документації на терміналах tty і X
grog	Читає файли і вгадує яка опція groff -e, -man, -me, -mm, ms, -p, -s і -t є необхідною для виводу файлів, і звітує команді groff включаючи ці опції
grolbp	Драйвер groff для принтерів Cannon CAPSL (серії LBP-4 і LBP-8 лазерних принтерів)
grolj4	Драйвер для groff, який створює вивід у придатному форматі PCL5 для принтерів HP LaserJet 4 printer
grops	Перекладає вивід команди GNU troff у PostScript
grotty	Перетворює вивід програми GNU troff у придатну форму для пристроїв подібних до друкарських машинок
gtbl	посилання до tbl
hpftodit	Стиворює файл шрифту для використання grof -Tlj4 з метричного файлу HP-tagget
indxbib	Створює інвертований список для бібліографічних баз даних з зазначених файлів для використання refer, lookbib і lkbib
lkbib	Шукає довідки у бібліографічних баз даних які вміщують вказані ключі і повідомляє знайдені
lookbib	Виводить підказку на стандартний потік помилок (якщо тільки стандартний потік вводу не є терміналом), читає рядки, які містять множину ключових слів з стандартного вводу, шукає довідки у бібліографічних баз даних у вказаному файлі, виводить будь-які знайдені довідки на стандартний вивід і повторяє цей процес поки не прийде кінець вводу

mmroff	Простий препроцесор для groff
neqn	Форматує рівняння для виводу ASCII (American Standart Code for Information Interchange)
nroff	Скрипт, який емулює команду nroff використовуючи groff
pdfroff	Створює pdf документи використовуючи groff
pfbtops	Перекладає шрифт PostScript у форматі .pfb у ASCII
pic	Компілює опис зображень вбудований файлми troff чи Tex у команди зрозумілі для Tex чи troff
pic2graph	Перетворює діаграми PIC у обрізані зображення
post-grohtml	Перетворює вивід команди GNU troff у HTML
preconv	Перетворює кодування ввідного файлу у щось, що розуміє GNU troff
pre-grohtml	Перетворює вивід GNU troff у HTML
refer	Копіює контент файлу у стандартний вивід, крім рядків між символами . [і .], які інтерпретуються як цитати, і рядків між .R1 і .R2, які інтерпретуються як команди, які вказують, як ці цитати обробляти
roff2dvi	Перетворює файли roff у формат DVI
roff2html	Перетворюють файли roff у формат HTML
roff2pdf	Перетворює фйли roff у файл PDF
roff2ps	Перетворює файли roff у файли ps
roff2text	Перетворює файли roff у текстові файли
roff2x	Перетворює файли roff у інші формати
soelim	Читає файли і замінює рядки формату .so file вмістом згаданого файлу file
tbl	Компілює опис таблиць вбудованих у файли вводу troff у команди, які зрозумілі для troff
tfmtodit	Створює файл шрифту які використовуються groff -Tdvi
troff	Високо сумісна утиліта з програмою troff; вона повинна викликатися використовуючи команду groff, яка також буде запускатися препроцесорами і постпроцесорами у відповідному порядку і з відповідними опціями

6.46. Xz-5.0.4

Пакет Xz вміщує програми для компресії і декомпресії файлів. Вона забезпечує можливості для форматів компресії lzma і новішого xz. Стискаючи текстові файли з xz дає кращий процент компресії ніж традиційні команди gzip чи bzip2.

Приблизний час побудови: 0.3 SBU
Необхідний дисковий простір: 18 Мбайт

6.46.1 Встановлення Xz

Підготуйте Xz до компіляції:

```
./configure --prefix=/usr --libdir=/lib —docdir=/usr/share/doc/xz-5.0.4
```

Скомпілюйте пакет:

```
make
```

Для тестування результатів, виконайте:

```
make check
```

Встановіть пакет:

```
make pkgconfigdir=/usr/lib/pkgconfig install
```

6.46.2 Вміст Xz

Встановлені програми: lzcat (link to xz), lzcmp (link to xzdiff), lzdiff (link to xzdiff), lzegrep (link to xzgrep), lzfgrep (link to xzgrep), lzgrep (link to xzgrep), lzless (link to xzless), lzma (link to xz), lzmadec, lzmainfo, lzmore (link to xzmore), unlzma (link to xz), unxz, (link to xz), xz, xzcat (link to xz), xzcmp (link to xzdiff), xzdec, xzdiff, xzegrep (link to xzgrep), xzfgrep (link to xzgrep), xzgrep, xzless, xzmore

Встановлені бібліотеки: liblzma.{a,so}

Встановлені директорії: /usr/include/lzma і /usr/share/doc/xz-5.0.4

Короткий опис

lzcat Виконує декомпресію в стандартний вивід

lzcmp Виконує cmp на стиснених файлах LZMA

lzdiff Виконує diff на стиснених файлах LZMA

lzegrep Виконує egrep на стиснених файлах LZMA

lzfgrep Виконує fgrep на стиснених файлах LZMA

lzgrep Виконує grep на стиснених файлах LZMA

lzless Виконує less на стиснених файлах LZMA

lzma	Стискає чи розпаковує файли використовуючи формат LZMA
lzmadec	Малий і швидкий декодер для стиснутих файлів LZMA
lzmainfo	Показує інформацію, яка збережена у заголовку стисненого файлу LZMA
lzmore	Виконує more на стиснених файлах LZMA
unlzma	Розпаковує файли використовуючи формат LZMA
unxz	Розпаковує файли використовуючи формат XZ
xz	Стискає чи розпаковує файли використовуючи формат XZ
xzcat	Розпаковує у стандартний вивід
xzcmp	Виконує cmp на стиснених файлах XZ
xzdec	Малий і швидкий декодер для стиснених файлах XZ
xzdiff	Виконує diff на стиснених файлах XZ
xzegrep	Виконує egrep на стиснених файлах XZ
xzfgrep	Виконує fgrep на стиснених файлах XZ
xzgrep	Виконує grep на стиснених файлах XZ
xzless	Виконує less на стиснених файлах XZ
xzmore	Виконує more на стиснених файлах XZ
liblzma*	Бібліотека, яка реалізує блок-сортуючу компресію без втрат, використовуючи ланцюговий алгоритм Лемпеля-Зіва-Велча

6.47. GRUB-2.00

Пакет GRUB вміщує Grand Unified Bootloader

Приблизний час побудови: 0.7 SBU
Необхідний дисковий простір: 112 Мбайт

6.47.1 Встановлення GRUB

Виправте несумісність між цим пакетом і Glibc-2.16.0:

```
sed -i -e '/gets is a/d' grub-core/gnulib/stdio.in.h
```

Підготуйте GRUB до компіляції:

```
./configure --prefix=/usr
--sysconfdir=/etc \
--disable-grub-emu-usb \
--disable-efiemu \
--disable-werror
```

Опція --disable-werror дозволяє побудові завершуватися з попередженнями введеними більш сучасною версією flex. Інші прапорці —disable мінімізують побудову, вимикаючи можливості і тестові програми, які не потрібні для LFS.

Скомпілюйте пакет:

```
make
```

Цей пакет не постачається з тестовим набором програм.

Встановіть пакет:

```
make install
```

Використовуючи GRUB для того, щоб зробити вашу систему LFS доступною для завантаження буде обговорено у Секції 8.4, “Використання GRUB для встановлення процесу завантаження”.

6.47.2 Вміст GRUB

Встановлені програми: grub-bin2h, grub-bios-setup, grub-editenv, grub-fstest, grub-install, grub-kbdcomp, grub-menulst2cfg, grub-mkconfig, grub-mkdevicemap, grub-mkimage, grub-mklayout, grub-mknetdir, grub-mkpasswd-pbkdf2, grub-mkrelpath, grub-mkrescue, grub-mkstandalone, grub-ofpathname, grub-probe, grub-reboot, grub-script-check, grub-set-default, grub-setup, grub-sparc64-setup

Встановлені директорії: /usr/lib/grub, /etc/grub.d, /usr/share/grub, /boot/grub

Короткий опис

grub-bin2h Перетворює бінарний файл у заголовкивий файл C

grub-bios-setup Допоміжна програма для grub-install

grub-editenv Утиліта для редагування блоку середовища

grub-fstest	Утиліта для відлагоджування драйверу файлової системи
grub-install	Встановлює GRUB на ваш пристрій
grub-kbdcomp	Скрипт, який перетворює рокладку xkb у таку, яка розпізнається GRUB
grub-menulst2cfg	Перетворює файл GRUB menu.lst у файл grub.cfg, який використовується GRUB2
grub-mkconfig	Генерує файл конфігурації grub
grub-mkdevicemap	Автоматично генерує файл, який містить карту пристрою
grub-mkimage	Створює образ GRUB, який має можливість завантаження
grub-layout	Генерує файл розкладки клавіатури GRUB
grub-mknetdir	Підготовлює директорію мережного завантаження GRUB
grub-mkpasswd-pbkdf2	Генерує зашифрований пароль PBKDF2 для використання у меню завантаження
grub-mkrelpath	Робить шлях до системи відносний до її кореневої директорії
grub-mkrescue	Робить завантажувальний образ GRUB доступним для дискет чи CD-ROM/DVD
grub-mkstandalone	Створює автономний образ
grub-ofpathname	Допоміжна програма, яка виводить шлях до пристрою GRUB
grub-probe	Визначає інформацію про пристрій для даного шляху чи пристрою
grub-rebot	Встановлює типовий завантажувальний вхід для GRUB тільки для наступного завантаження
grub-script-check	Перевіряє конфігураційний скрипт GRUB на предмет синтаксичних помилок
grub-set-default	Встановлює типовий завантажувальний вхід для GRUB
grub-setup	Встановлює образ для завантаження з пристрою
grub-sparc64-setup	Допоміжна програма для grub-setup

6.48 Less-444

Пакет Less вміщує переглядач текстових файлів.

Приблизний час побудови: менш ніж 0.1 SBU
Необхідний дисковий простір: 3.8 Мбайт

6.48.1 Встановлення Less

Підготуйте less до встановлення:

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

Значення опцій конфігурування:

`--sysconfdir=/etc`

Ця опція вказує програмам створеним цим пакетом шукати конфігураційні файли у каталозі /etc.

Скомпілюйте пакет:

```
make
```

Цей пакет не постачається з тестовим набором програм.

Встановіть пакет:

```
make install
```

6.48.2 Вміст Less

Встановлені програми: less, lessecho і lesskey

Короткий опис

less	Переглядач файлів; він виводить вміст вказаного файлу, дозволяючи користувачу прокручувати, знаходити рядки і переходити до міток
lessecho	Необхідна для розширення мета-символів, такі як * і ?, у назвах файлів систем Юнікс
lesskey	Використовується для специфікації прив'язки ключів для less

6.49. Gzip-1.5

Пакет Gzip вміщує програми для стиснення і розпаковування файлів.

Приблизний час побудови: 0.2 SBU
Необхідний дисковий простір: 19.7 Мбайт

6.49.1 Встановлення Gzip

Підготуйте Gzip до компіляції:

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

Скомпілюйте пакет:

```
make
```

Для тестування результатів, виконайте:

```
make check
```

Встановіть пакет:

```
make install
```

Перемістіть деякі програми, які необов'язкові для кореневої файлової системи:

```
mv -v /bin/{gzexe,uncompress,zcmp,zdiff,zegrep} /usr/bin
mv -v /bin/{zfgrep,zforce,zgrep,zless,zmore,znew} /usr/bin
```

6.49.2 Вміст Gzip

Встановлені програми: gunzip, gzexe, gzip, uncompress, zcat, zcmp, zdiff, zegrep, zfgrep, zforce, zgrep, zless, zmore і znew

Короткий опис

gunzip Розпаковує стиснені файли

gzexe Створює виконувані файли, які саморозпаковуються

gzip Стискує дані фвйли використовуючи кодування Лемпеля-Зіва

uncompress Розпаковує стиснені файли

zcat Розпаковує дані стиснені файли у стандартний вивід

zcmp Виконує cmp на стиснених файлах

zdiff Виконує diff на стиснених файлах

zegrep Виконує egrep на стиснених файлах

zfgrep Виконує fgrep на стиснених файлах

zforce Застосовує розширення .gz на усі вказані файли, які є стиснуті за

	допомогою gzip, отже gzip не буде стискувати їх знову; це може бути корисним коли імена файлів були обрізаними під час передачі
zgrep	Виконує grep на стиснутих файлах
zless	Виконує less на стиснутих файлах
zmore	Виконує more на стиснутих файлах
znew	Стискає повторно файли з формату compress у формат gzip — з .Z у .gz

6.50 IPRoute2-3.5.1

Пакет IPRoute2 вміщує програми для базових і розширених мереж IPv4.

Приблизний час побудови: 0.1 SBU
Необхідний дисковий простір: 7.3 Мбайт

6.50.1 Встановлення IPRoute2

Бінарний файл arpd, який міститься у цьому пакеті, залежить від Berkeley DB. Через те, що arpd не є загально потрібним для базових систем Лінукс, усуньте залежність від Berkeley DB виконуючи команди розміщені нижче. Якщо бінарний файл arpd є необхіддним, інструкції для компілювання Berkeley DB можуть бути знайденими у книзі BLFS за адресою *http://www.linuxfromscratch.org/blfs/view/svn/server/databases.html#db.*

```
sed -i '/^TARGETS/s@arpd@@g' misc/Makefile
sed -i /ARPD/d Makefile
sed -i 's/arpd.8//' man/man8/Makefile
```

Скомпілюйте пакет:

```
make DESTDIR=
```

Значення опцій make:

DESTDIR=

Це запевняє, що бінарні файли IPRoute2 будуть встановленими у коректу директорію. Типово, DESTDIR встановлена у `/usr`.

Цей пакет постачається з тестовим набором, але через міри прийняті у ньому, стає неможливо надійно виконувати ці тести з середовища chroot. Якщо ви бажаєте виконати ці тести після завантаження у вашу нову систему LFS, впевніться, що ви вибрали підтримку `/proc/config.gz CONFIG_IKCONFIG_PROC` (“General setup” -> “Enable access to .config through /proc/config.gz”) у вашому ядрі, після цього виконайте 'make alltestss' з субдиректорії `testsuite/`.

Встановіть пакет:

```
make DESTDIR= \
MANDIR=/usr/share/man \
DOCDIR=/usr/share/doc/iproute2-3.5.1 install
```

6.50.2 Вміст IPRoute2

Встановлені програми: ctstat (link to lnstat), genl, ifcfg, ifstat, ip, lnstat, nstat, routef, routel, rtacct, rtmon, rtpr, rtstat (link to lnstat), ss і tc

Встановлені директорії: /etc/iproute2, /lib/tc, /usr/share/doc/iproute2-3.5.1, /usr/lib/tc

Короткий опис

ctstat Утиліта статусів з'єднання

ifcfg Скрипт-обгортка оболонки для команди ip. Зауважте, що вона вимагає програми arping і rdisk з пакету iputils, які можуть бути знайденими за

адресою http://www.skbuff.net/iputils/.

ifstat Показує статистику інтерфейсу, включаючи кількість переданих і отриманих пакетів через інтерфейс

ip Головна програма. Вона має декілька різних функцій:

- ip link <пристрій> дозволяє користувачам дивитись на стан пристрою і виконувати зміни
- ip addr дозволяє користувачам переглядати адреси і їхні всластивості, додавати нові адреси і видаляти старі
- ip neighbor дозволяє користувачам переглядати сусідні зв'язки і їхні властивості, додавати нові сусідні входження і видаляти старі
- ip rule дозволяє користувачам переглядати політики маршрутизації і змінювати їх
- ip route дозволяє користувачам переглядати таблицю маршрутизації і змінювати правила таблиць маршрутизації
- ip tunnel дозволяє користувачам переглядати IP тунелі і їхні властивості і змінювати їх
- ip maddr дозволяє користувачам переглядати багатоскладові адреси і їхні властивості і змінювати їх
- ip mroute дозволяє користувачам встановлювати, змінювати чи видаляти багатоадресну маршрутизацію
- ip monitor дозволяє користувачам слідкувати у поточному часі стан пристроїв, адрес і маршрути

lnstat Показує статистику мережі Лінукс. Вона є узагальненою і більш функціональною заміною для старої програми rtstat

nstat Показує статистику мережі

routef Компонент ip route. Вона використовується для скидання таблиць маршрутизації

routel Компонент ip route. Вона використовується для перераховування таблиць маршрутизації

rtacct Виводить вміст /proc/net/rt_acct

rtmon Утиліта для відслідковування маршрутизації

rtpr Перетворює вивід програми ip -o назад у читабельну форму

rtstat Утиліта статусу маршрутизації

ss Команда похожа до програми netstat; показує активні підключення

tc Програма для контролю трафіку; вона реалізує QOS (Quality Of Service — якість сервісу) і COS (Class Of Service — клас сервісу)

- tc qdisc дозволяє користувачам встановлювати дисципліну масового обслуговування
- tc class Дозволяє користувачам встановлювати класи засновані на плануванні дисциплін масового обслуговування
- tc estimator дозволяє користувачам оцінювати мережний потік
- tc filter дозволяє користувачам встановлювати фільтрування пакетів QOS/COS
- tc policy Дозволяє користувачам встановлювати політики QOS/COS

6.51. Kdb-1.15.3

Пакет Kdb вміщує файли ключ-таблиць, консольні шрифти і клавіатурні утиліти

Приблизний час побудови: 0.1 SBU
Необхідний дисковий простір: 20 Мбайт

6.51.1 Встановлення Kdb

Поведінка loadkeys при виклику без імені файлу є неправильною у цьому випуску. Наступний патч виправить це:

```
patch -Np1 -i ../kdb-1.15.3-upstream_fixes-1.patch
```

Поведінка клавіш Backspace і Delete не сумісна між картами клавіш у пакеті Kdb. Наступний патч виправить цю проблему для карт клавіш i386:

```
patch -Np1 -i ../kbd-1.15.3-backspace-1.patch
```

Після використання патчів, клавіша Backspace генерує символ з кодом 127, і клавіша Delete генерує добре відому послідовність.

Виправте помилку програми у перекладах Іспанських повідомлень, що запобігає побудову kdb з gettext-0.18.1.1:

```
sed -i '/guardado\ el/s/\(^.*en\ %\)\(.*\)/\14\$\2/' po/es.po
```

Видаліть зайву програму resizecons (тільки для 32-бітних архітектур x86, потребує не існуючу svgalib, що передує linux-2.6 і є несумісною з сучасними KMS, для забезпечення файлів відео режимів — використовуйте setfont, яка встановлює розмір консолі належним чином і його сторінку документації.

Новіша версія configure.ac є новішою ніж aclocal.m4, отож autotools будуть виконуватися програмою make, яка перезапише зміну до configure. Замініть дату і час файлу configure.ac — це використовує системний часовий пояс, отож ми обираємо час, який достатньо старіший, ніж усі часові зони.

```
sed -i 's/\(RESIZECONS_PROGS=\)yes/\1no/' configure &&
sed -i 's/resizecons.8 //' man/man8/Makefile.in &&
touch -d '2011-05-07 08:30' configure.ac
```

Підготуйте Kdb до компіляції:

```
./configure --prefix=/usr –datadir=/lib/kbd
```

Значення опцій конфігурування:

`--datadir=/lib/kdb`

Цей прапорець кладе дані розкладок клавіатур у директорії, яка завжди буде на кореневому розділі, взамін каталогу за замовчуванням /usr/share/kdb.

Скомпілюйте пакет:

```
make
```

Цей пакет не постачається з тестовим набором.

Встановіть пакет:

```
make install
```

Увага

Для деяких мов (такі як Білоруська) пакет Kdb не постачає корисну розкладку клавіатури, де сток розкладку клавіатури припускає кодування ISO-8859-5 і використовується розкладка клавіатури CP1251. Користувачі таких мов мають завантажити працюючі розкладки клавіатур окремо.

Деякі скрипти у пакеті LFS-Bootscripts залежать від kdb_mode, loadkeys, openvt і setfont. Так як каталог /usr може не бути доступним під час ранніх стадій завантаження системи, ці бінарні файли необхідно перемістити на кореневий розділ:

```
mv -v /usr/bin/{kbd_mode,loadkeys,openvt,setfont} /bin
```

За бажанням, встановіть документацію:

```
mkdir -v /usr/share/doc/kbd-1.15.3
cp -R -v doc/* \
/usr/share/doc/kbd-1.15.3
```

6.51.2 Вміст Kdb

Встановлені програми: chvt, deallocvt, dumpkeys, fgconsole, getkeycodes, kbd_mode, kbdrate, loadkeys, loadunimap, mapscrn, openvt, psfaddtable (link to psfxtable), psfgettable (link to psfxtable), psfstriptable (link to psfxtable), psfxtable, setfont, setkeycodes, setleds, setmetamode, showconsolefont, showkey, unicode_start, and unicode_stop

Встановлені директорії: /lib/kdb

Короткий опис

chvt Змінює передній план віртуального терміналу

deallocvt Звільняє невикористовувані віртуальні термінали

dumpkeys Робить дамп таблиці перекладів клавіатур

fgconsole Виводить кількість активних віртуальних терміналів

getkeycodes Виводить таблицю ядра, яка показує коду клавіші у відповідності до сканкоду

kdb_mode Повідомляє чи встановлює режим клавіатури

kdbrate Встановлює швидкості повтору клавіатури і її затримку

loadkeys Завантажує таблиці перекладів клавіатури

loadunimap Завантажує таблицю перетворення юнікоду у шрифт

mapscrn Застаріла програма, яка використовується для завантаження визначених користувачем таблицю символів виводу у консольний драйвер; тепер це робить setfont

openvt	Запускає програму на новому віртуальному драйвері (VT)
psfaddtable	Посилання до psfxtable
psfgettable	Посилання до psfxtable
psstriptable	Посилання до psfxtable
psfxtable	Оборбляє таблиці символів юнікод для конольних шрифтів
setfont	Змінює шрифти EGA і VGA на консольні
setkeycodes	Завантажує таблиці ядра перетворень з сканкодів у коди клавіш; вона корисна коли існують нестандартні клавіші на клавіатурі
setleds	Встановлює прапорці клавіатури і її LED'и (Light Emitting Diodes)
setmetamode	Визначають обробку мета-клавіш клавіатури
showconsolefont	Показує поточний EGA/VGA консольний шрифт екрану
showkey	Повідомляє сканкоди, коди клавіш і коди ASCII клавіш натиснутих на клавіатурі
unicode_start	Встановлює клавіатуру і консоль у режим UNICODE. Не використовуйте цю програму, за винятком, якщо ваш файл клавіатури є у кодуванні ISO-8859-1. Для інших кодувань, ця утиліта продукує некоректні результати.
unicode_stop	Повертає клавіатуру і консоль з режиму UNICODE

6.52. Kmod-9

Пакет Kmod містить бібліотеки і утиліти для завантаження модулів ядра.

Приблизний час побудови: 0.1 SBU
Необхідний дисковий простір: 30 Мбайт

6.52.1 Встановлення Kmod

Застосуйте наступний патч для виправлення проблеми під час виконання тестового набору на хостах x86:

```
patch -Np1 -i ../kmod-9-testsuite-1.patch
```

Підготуйте Kmod до компіляції:

```
./configure —prefix=/usr \
--bindir=/bin \
--libdir=/lib \
--sysconfdir=/etc \
--with-xz \
--with-zlib
```

Значення опцій конфігурування:

lib і --with-**

Ці опції вмикають підтримку у Kmod оброблення стиснутих модулів ядра. Змінні оболонки є необхідними за порядком знаходження заголовкових файлів xz і zlib і бібліотек за відсутності pkg-config.

Скомпілюйте пакет:

```
make
```

Для тестування результатів, виконайте:

```
make check
```

Встановіть пакет, і створіть посилання для сумісності з Module-Init-Tools, пакет, який попередньо обробляв модулі ядра Лінукс:

```
make pkgconfigdir=/usr/lib/pkgconfig install

for target in depmod insmod modinfo modprobe rmmod; do
 ln -sv ../bin/kmod /sbin/$target
done

ln -sv kmod /bin/lsmod
```

6.52.2 Вміст Kmod

Встановлені програми: depmod (link to kmod), insmod (link to kmod), kmod, kmod-nolib, lsmod (link до kmod), modinfo (link to kmod), modprobe (link to kmod) і rmmod (link to kmod)

Встановлені бібліотеки: /lib/kmod.so

Короткий опис

depmod	Створює файл залежності заснований на символах, які вона знайде у існуючому наборі модулів; файл залежності використовується командою modprobe для автоматичного завантаження необхідних модулів
insmod	Встановлює доступні до завантаження модулі у працююче ядро
libkmod	Ця бібліотека використовується іншими програмами для завантаження і вивантаження модулів ядра
lsmod	Перераховує поточно завантажені модулі
modinfo	Перевіряє об'єктний файл асоційований з модулем ядра і виводить будь-яку інформацію, яку він може назбирати
modprobe	Використовує файл залежності, створеним depmod, для автоматичного завантаження необхідних модулів
rmmod	Вивантажує модулі з працюючого ядра

6.53. Libpipeline-1.2.1

Пакет Libpipeline містить бібліотеку для маніпулювання шлюзів підпроцесів у гнучкий і зручний спосіб.

Приблизний час побудови: 0.2 SBU
Необхідний дисковий простір: 7.4 Мбайт

6.53.1. Встновлення Libpipeline

Виправте несумісність між цим пакетом і Glibc-2.16.0:

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

Підготуйте Libpipeline до компіляції:

```
PKG_CONFIG_PATH=/tools/lib/pkgconfig ./configure --prefix=/usr
```

Значення опцій конфігурування:

PKG_CONFIG_PATH

Використовує pkg-config для отримання місцезнаходження метаданих тестової бібліотеки побудованої у Секції 5.13, “Check-0.9.8”.

Скомпілйте пакет:

```
make
```

Для тестування результатів, виконайте:

```
make check
```

Встановіть пакет:

```
make install
```

6.53.2 Вміст Libpipeline

Встановлені бібліотеки: libpipeline.so

Короткий опис

libpipeline Ця бібліотека використовується для безпечного створення шлюзів між процесами

6.54. Make-3.82

Пакет Make містить програми для компіляції пакетів.

Приблизний час побудови: 0.4 SBU
Необхідний дисковий простір: 11.3 Мбайт

6.54.1 Встановлення Make

Для початку застосуйте деякі патчі потоку:

```
patch -Np1 -i ../make-3.82-upstream_fixes-2.patch
```

Підготуйте Make до компіляції:

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

Скомпілюйте пакет

```
make
```

Для тестування результатів, виконайте:

```
make check
```

Встановіть пакет:

```
make install
```

6.54.2 Вміст Make

Встановлені програми: make

Короткий опис

make Автоматично визначає які куски пакету потрібно скомпілювати і тоді виконує необхідні команди

6.55 Man-DB-2.6.2

Пакет Man-DB містить програми для знаходження і переглядання сторінок документації.

Приблизний час побудови: 0.5 SBU
Необхідний дисковий простір: 27 Мбайт

6.55.1 Встановлення Man-DB

Виправте несумісність між цим пакетом і Glibc-2.16.0

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

Підготуйте Man-DB до компіляції

```
./configure —prefix=/usr \
--libexecdir=/usr/lib \
--docdir=/usr/share/doc/man-db-2.6.2 \
--sysconfdir=/etc \
--disable-setuid \
--with-browser=/usr/bin/lynx \
--with-vgrind=/usr/bin/vgrind \
--with-grap=/usr/bin/grap
```

Значення опцій конфігурування:

`--disable-setuid`

Цей прапорець вимикає побудову setuid програми man для користувача man

`--with-...`

Ці три параметри використовуються для встановлення типових програм. Lynx є текстовий веб переглядач (перегляньте BLFS для інструкцій по встановленню), vgrind перетворює код програм у ввід Groff і grap є корисною для встановлення типів графів у документах Groff. Програми vgrind і grap зазвичай не потрібні для переглядання сторінок документації. Вони не є частиною LFS чи BLFS, але ви повині б встановити їх після закінчення LFS.

Скомпілюйте пакет:

```
make
```

Для тестування результатів, виконайте:

```
make check
```

Встановіть пакет:

```
make install
```

6.55.2 Сторінки допомогою відмінних від англійської у LFS

Наступні таблиці показують набір символів, що спричиняють встановлення сторінок документації Man-DB під **/usr/share/man/<ll>** у яких вони будуть кодуватися.

Мова (код)	**Кодування**	**Мова (код)**	**Кодування**
Данська (da)	ISO-8859-1	Хорватська (hr)	ISO-8859-2

Німецька (de)	ISO-8859-1	Угорська (hu)	ISO-8859-2
Англійська (en)	ISO-8859-1	Японська (ja)	EUC-JP
Іспанська (es)	ISO-8859-1	Корейська (ko)	EUC-KR
Естонська (et)	ISO-8859-1	Литовська (lt)	ISO-8859-13
Фінська (fi)	ISO-8859-1	Латвійська (lv)	ISO-8859-13
Французька (fr)	ISO-8859-1	Македонська (mk)	ISO-8859-5
Ірландська (ga)	ISO-8859-1	Польська (pl)	ISO-8859-2
Голійська (gl)	ISO-8859-1	Румунська (ro)	ISO-8859-2
Індонезійська (id)	ISO-8859-1	Російська (ru)	KOI8-R
Ісландська (is)	ISO-8859-1	Словацька (sk)	ISO-8859-2
Італіська (it)	ISO-8859-1	Словенська (sl)	ISO-8859-2
Норвезький букмол (nb)	ISO-8859-1	Сарбська латиниця (sr@latin)	ISO-8859-2
Голондська (nl)	ISO-8859-1	Сербська (sr)	ISO-8859-5
Норвезький нюнорськ (nn)	ISO-8859-1	Турецька (tr)	ISO-8859-9
Норвезька (no)	ISO-8859-1	Українська (uk)	KOI8-U
Португальська (pt)	ISO-8859-1	В'єтнамська (vi)	TCVN5712-1
Шведська (sv)	ISO-8859-1	Спрощена Китайська (zh_CH)	GBK
Білоруська (be)	CP1251	Спрощена Китайська, Сингапур (zh_SG)	GBK
Болгарська (bg)	CP1251	Традиційна Китайська, Гог Конг (zh_HK)	BIG5HKSCS
Чеська (cs)	ISO-8859-2	Традиційна китайська (zh_TW)	BIG5
Грецька (el)	ISO-8859-7		

Увага

Сторінки допомоги у мовах, які не перераховані тут, не підтримуються.

6.55.3 Вміст Man-DB

Встановлені програми: accessdb, apropos (link to whatis), catman, lexgrog, man, mandb, manpath, whatis, і zsoelim

Встановлені директорії: /usr/lib/man-db, /usr/share/doc/man-db

Короткий опис

accessdb Робить дамп вмісту бази даних whatis у читабельній для людини формі

apropos Шукає базу даних whatis і виводить короткий опис системних команд, які вміщує даний рядок

catman Створює або оновлює передформатовані сторінки допомоги

lexgrog Виводить рядок сумарної інформації про дану сторінку допомоги

man Форматує і виводить вказану сторінку допомоги

mandb Створює або оновлює базу даних whatis

manpath Виводить вміст $MANPATH чи (якщо $MANPATH не встановлена) відповідний шлях пошуку заснований на рядках у man.conf і середовищі користувача

whatis Ierfбазу даних whatis і виводить короткий опис системних команд які містять дане ключ як окреме слово

zsoelim Читає файли і замінює рядки, які мають форму .so file вмістом згаданого файлу file

6.56. Patch-2.6.1

Пакет Patch вміщує програму для модифікування чи створення файлів застосуванням файлу “patch”, який типово створюється програмою diff.

Приблизний час побудови: менш ніж 0.1 SBU
Необхідний дисковий простір: 3,4 Мбайт

6.56.1 Встановлення Patch

Застосуйте патч для запобігання виконання тестового набору, який вимагає ed:

```
patch -Np1 -i ../patch-2.6.1-test_fix-1.patch
```

Підготуйте Patch до компіляції:

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

Скомпілюйте пакет:

```
make
```

Для тестування результатів, виконайте:

```
make -k check
```

Встановіть пакет:

```
make install
```

6.56.2. Вміст Patch

Встановлена програма: patch

Короткий опис

patch Модифікує файли у відповідності до файлу патчу. Патч файл типово є списком різниць між файлами створений з програмою diff. Застосовуючи ці різниці до оригінальних файлів, patch створює пропатчені версії.

6.57. Sysklogd-1.5

Пакет Sysklogd вміщує програми для журналювання системних повідомлень, такі як згенеровані ядром під час надзвичайних подій.

Приблизний час побудови: менш ніж 0.1 SBU
Необхідний дисковий простір: 0.6 Мбайт

6.57.1. Встановлення Sysklogd

Скомпілюйте пакет:

```
make
```

Цей пакет не постачається з тестовим набором.

Встановіть пакет:

```
make BINDIR=/sbin install
```

6.57.2 Конфігурування Sysklogd

Створіть новий файл `/etc/syslog.conf` виконуючи наступну команду:

```
cat > /etc/syslog.conf << "EOF"
# Begin /etc/syslog.conf
auth,authpriv.* -/var/log/auth.log
*.*;auth,authpriv.немає -/var/log/sys.log
daemon.* -/var/log/daemon.log
kern.* -/var/log/kern.log
mail.* -/var/log/mail.log
user.* -/var/log/user.log
*.emerg *
# End /etc/syslog.conf
EOF
```

6.57.3. Вміст Sysklogd

Встановлені програми: klog і syslogd

Короткий опис

klogd Системний демон для перехоплення повідомлень ядра

syslogd Записує в файл журналу повідомлення, які генерують системні програми. Кожне записане повідомлення містить як мінімум дату і ім'я хосту, і зазвичай ім'я програми, але це залежить від того як сконфігурований демон.

6.58. Sysvinit-2.88dsf

Пактет Sysvinit вміщує програми для контролювання старту, виконання і вимкнення системи.

Приблизний час побудови: менш ніж 0.1 SBU
Необхідний дисковий простір: 1.4 Мбайт

6.58.1. Встановлення Sysvinit

Коли рівень виконання змінено (для прикладу, при вимкненні системи), **init** посилає сигнали завершення до тих процесів, які **init** запустив і які не повинні працювати у новому рівні виконання. Під час виконання цього, init виводить повідомлення як “Sending processes the TERM signal” (“Посилаю процесам сигнал TERM”), що означає вона посилає сигнали до усіх поточних процесів. Для запобігання неправильного розуміння цих повідомлень, модифікуйте код так, щоб ці повідомлення читались як “Sending processes configured via /etc/inittab the TERM signal ” (“Посилаю процесам сконфігурованим у `/etc/inittab` сигнал TERM”):

```
sed -i 's@Sending processes@& configured via /etc/inittab@g' src/init.c
```

Підтримувані версії програм **wall** і **mountpoint** були встановлені раніше пакетом Util-linux. Забороніть встановлення версій цих програм Sysinit і їхніх сторінок допомоги:

```
sed -i -e 's/utmpdump wall/utmpdump/' \
-e '/= mountpoint/d' \
-e 's/mountpoint.1 wall.1//' src/Makefile
```

Скомпілюйте пакет

```
make -C src
```

Цей пакет не постачається з тестовим набором.

Встановіть пакет:

```
make -C src install
```

6.58.2 Вміст Sysvinit

Встановлені програми: bootlogd, fstab-decode, halt, init, killall5, last, lastb (link to last), mesg, pidof (link to killall5), poweroff (link to halt), reboot (link to halt), runlevel, shutdown, sulogin, telinit (link to init), and utmpdump

Короткий опис

bootlogd Робить запис повідомлень завантаження до файлу журналу

fstab-decode Виконує команду з аргументами fstab-encoded

halt Типово викликає shutdown з опціями -h, за виключенням тих ситуацій, коли система вже у нульовому рівні виконанні, тоді вона вказує ядра вимкнути систему; вона записує у файл /vat/log/wtmp, що система вимкнута

init Перший процес, який запускається коли ядро ініціалізувало апаратне забезпечення і перебирає процес завантаження в свої руки щоб запустити процеси, які були їй вказані

killall5	Посилає сгнали до усіх процесів, за винятком процесів у її власній сесії, отож вона не вб'є оболонку в якій вона виконується
last	Показує які користувачі останніми ввійшли в систему (і вийшли), перебираючи записи у файлі /var/log/wtmp; вона також показує завантаження системи, вимкнення і зміни рівнів виконанням
lastb	Показує провали у входження в систему, які записані у /var/log/btmp
mesg	Контролює які користувачі можуть посилати повідомлення до поточного терміналу користувача
pidof	Повідомляє PID вказаних програм
poweroff	Вказує ядру вимкнути систему і вимкнути комп'ютер (дивіться halt)
reboot	Вказує ядру перевантажити систему (дивіться halt)
runlevel	Повідомляє попередній і поточний рівень виконання, який був занотований у файлі /var/run/utmp
shutdown	Вимикає систему у безпечний шлях, сигналізуючи усі процеси і повідомляючи усіх поточних користувачів системи
sulogin	Дозволяє користувачу root ввійти в систему; типово викликається програмою init коли система входить в режим одного користувача
telinit	Вказує init у який рівень виконання необхідно ввійти
utmpdump	Виводить вміст даного файлу журналу входу в систему у більш дружнім форматі для користувачів

6.59. Tar-1.26

Пакет Tar вміщує програму архіватор.

Приблизний час побудови: 2.4 SBU
Необхідний дисковий простір: 34 Мбайт

6.59.1. Встановлення Tar

Виправте несумісність між цим пакетом і Glibc-2.16.0:

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

Підготуйте Tar до компіляції:

```
FORCE_UNSAFE_CONFIGURE=1 \
./configure --prefix=/usr \
--bindir=/bin \
--libexecdir=/usr/sbin
```

Значення опцій конфігурування:

FORCE_UNSAFE_CONFIGURE=1

Це примушує тест для `mknod` виконувати від імені користувача root. Це зазвичай вважається небезпечним виконувати цей тест від імені користувача root, але він виконується на незавершеній системі, отже все гаразд.

Скомпілюйте пакет:

```
make
```

Для тестування результатів (приблизно 1 SBU), виконайте:

```
make check
```

Встановіть пакет:

```
make install
make -C doc install-html docdir=/usr/share/doc/tar-1.26
```

6.59.2. Вміст Tar

Встановлені програми: rmt і tar

Встановлені директорії: /usr/share/doc/tar-1.26

Короткий опис

rmt Віддалено маніпулює привід магнітної стрічки крізь міжпроцесне комунікаційне з'єднання

tar Створює, розархівовує файли і перераховує вміст архівів, також відомих як tarball

6.60 Texinfo-4.12a

Пакет Texinfo вміщує програми для читанняб запису і перетворення сторінок допомоги.

Приблизний час побудови: 0.2 SBU
Необхідний дисковий простір: 24 Мбайт

6.60.1. Встановлення Texinfo

Підготуйте Texinfo до компіляції:

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

Скомпілюйте пакет:

```
make
```

Для тестування результатів, виконайте:

```
make check
```

Встановіть пакет:

```
make install
```

Опціонально, встановіть компоненти які належать до процесу встановлення Tex

```
make TEXMF=/usr/share/texmf install-tex
```

Значення параметрів make:

```
TEXMF=/usr/share/texmf
```

Змінна файлу Makefile TEXMF вміщує місце знаходження кореневої папки пакету TeX якщо, для прикладу, пакет TeX буде встановлено пізніше.

Система документації Info використовує звичайний текстовий файл для розміщення його списку меню. Файл розміщений у `/usr/share/info/dir`. На жаль, через проблеми з версіями пакетів у Makefile, вона інколи має невідповідність з сторінками допомоги, встановленими у системі. Якщо коли-небудь виникне необхідність оновити файл `/usr/share/info/dir`, наступні опціональні команди виконають це завдання:

```
cd /usr/share/info
rm -v dir
for f in *
do install-info $f dir 2>/dev/null
done
```

6.60.2 Вміст Texinfo

Встановлені програми: info, infokey, install-info, makeinfo, pdftexi2dvi, texi2dvi, texi2pdf і texindex

Встановлені директорії: /usr/share/texinfo

Короткий опис

info Використовується для перечитування сторінок info які є аналогічні до сторінок man, але які найчастіше розкривають тему набагато глибше ніж просто роз'яснюючи усі доступні опції командного рядка. Для прикладу, порівняйте man bison і info bison

infokey	Компілює вихідний файл коду, який містить синтаксис Info у бінарний формат
install-info	Використовується для встановлення сторінок info; вона оновлює усі входження індексного файлу info
makeinfo	Перекладає даний файл Texinfo у сторінку info, звичайний текс чи HTML
pdftexi2dvi	Використовується для форматування даного документу Texinfo у файл PDF.
texi2dvi	Використовується для форматування даного документу Texinfo у файл незалежний від устаткування, який може бути роздрукований
texi2pdf	Використовується для форматування даного файлу Texinfo у файл PDF
texindex	Використовується для сортування індексних файлів Texinfo

6.61. Udev-118 (Витягнутий з systemd-188)

Пакет Udev містить програми для динамічного створення файлів пристроїв. Розроблення udev обєднано з systemd, але systemd є несумісним з LFS. Тут ми побудуємо і встановимо тільки необхідні файли udev.

Приблизний час побудови: 0.1 SBU
Необхідний дисковий простір: 17.4 Мбайт

6.61.1. Встановлення Udev

> Увага
>
> Цей пакет дещо відрізняється від інших. Початковий розпакований пакет — systemd-188.tar.xz, навіть якщо ми встановлюємо udev. Після зміни системної директорії, дотримуйтесь наступних інструкцій.

Архів (tarball) udev-lfs містить файли специфіковані для LFS для побудови Udev. Розпакуйте їх у директорію дерева коду systemd:

```
tar -xvf ../udev-lfs-188-3.tar.bz2
```

Скомпілюйте пакет:

```
make -f udev-lfs-188/Makefile.lfs
```

Встановіть програму:

```
make -f udev-lfs-188/Makefile.lfs install
```

На кінець встановіть стійкі мережні правила udev. Це завдання буде детально роз'яснено у Секції 7.2.1 “Створення стабільних імен для мережних інтерфейсів”. Зверніть увагу на те, що файлові системи /sys і /proc повинні бути змонтованими у середовищі створеним chroot як це було роз'яснено на початку цієї частини для наступної роботи скрипту.

```
bash udev-lfs-188/init-net-rules.sh
```

6.61.2. Вміст Udev

Встановлені програми:	accelerometer, ata_id, cdrom_id, collect, mtd_probe, scsi_id, v4l_id, udevadm, і udevd
Встановлені бібліотеки:	libudev.so
Встановлені директорії:	/etc/udev, /lib/udev, /lib/firmware, /usr/share/doc/udev

Короткий опис

ata_id	Забезпечує Udev унікальним рядком і додатковою інформацією (uuid,label) для пристроїв ATA
cdrom_id	Забезпечує Udev можливостями пристроїв CD-ROM чи DVD-ROM
collect	Дає ID для поточної події udev (uevent) і список ідентифікаторів (для усіх цільових подій udev — uevent), реєструє поточне ID і показує коли усі

	ідентифікатори були зареєстрованими
scsi_id	Забезпечує Udev унікальним ідентифікатором SCSI, основаним на даних повернених від посилання команди SCSI INQUIRY до вказаного пристрою
udevadm	Загальна утиліта адміністратора udev: контролює демон udev, забезпечує інформацію з бази даних Udev, виконує моніторинг uevents, чекає закінчення uevents, тестує конфігурацію Udev і генерує події для даного пристрою
udevd	Демон, який уловлює uevents на сокеті netlink, створює пристрої і виконує сконфігуровані зовнішні програми у відповідь на ці події udev (uevents)
`libudev`	Бібліотека інтерфейсу до інформації про пристрої udev
`/etc/udev`	Містить конфігураційні файли Udev, права доступу до пристроїв і правила для іменування пристроїв

6.62. Vim-7.3

Пакет Vim містить потужний текстовий редактор.

Приблизний час побудови: 1.1 SBU
Необхідний дисковий простір: 96 Мбайт

> Альтернативи до Vim
>
> Якщо віддаєте перевагу іншому редактору — такому як Emacs, Joe чи Nano — будь-ласка перегляньте відповідні інструкції до встановлення за адресою http://www.linuxfromscratch.org/blfs/view/svn/postlfs/editors.html

6.62.1. Встановлення Vim

Для початку, змініть типове розміщення конфігураційного файлу vimrc на /etc:

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

Тепер підготуйте Vim до компіляції

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

Значення опцій конфігурування:

`--enable-multibyte`

Цей прапорець вмикає підтримку редагування файлів з багатобайтними символьними кодуваннями. Це необхідно при використанні локаль з багатобайтним символьним набором. Цей прапорець є також корисним для можливості редагування текстових файлів створених дистрибутивами, такими як Fedora, які використовують UTF-8 як типовий символьний набір.

Скомпілюйте пакет:

```
make
```

Для тестування результатів, виконайте:

```
make test
```

Однак, цей тестовий набір виводить велику кількість бінарних даних на екран, що може спричинити проблеми з налаштуваннями поточного терміналу. Це може бути вирішеним перенаправленням виводу у файл журналу. Успішний тест закінчується словами “ALL DONE”.

Встановіть пакет:

```
make install
```

Багато користувачів використовують vi замість vim. Для дозволу виконання vim коли користувачі звично вводять vi, створіть посилання до бінарного файлу і сторінки допомоги в доступних мовах:

```
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
```

За звичай, документація Vim встановлюється у директорії /usr/share/vim. Наступні посилання дозволяють доступ до документації за допомогою /usr/share/doc/vim-7.3, роблячи її сумісною з місцезнаходженням документації інших пакетів:

```
ln -sv ../vim/vim73/doc /usr/share/doc/vim-7.3
```

Якщо на систему LFS буде встановлюватися X Window System, може бути необхідність скомпілювати ще раз Vim після встановлення X. Vim постачається з версією GUI редактору, який вимагає X і деякі додаткові бібліотеки до встановлення. Для докладнішої інформації цього процесу, перегляньте документацію Vim і сторінку встановлення Vim у книзі BLFS за адресою *http://www.linuxfromscratch.org/blfs/view/svn/postlfs/editors.html#postlfs-editors-vim.*

6.62.2. Конфігурація Vim

За звичай, **vim** виконується у режимі несумісному з vi. Це може бути новим для користувачів які використовували інші редактори у минулому. Опція “nocompatible” додана нижче для підкреслення факту використання нової поведінки. Вона також нагадує тим, хто забажає змінити її на режим “compatible” і що це опція, яка повинна стояти на початку файлу конфігурації. Це є необхідним через те, що вона змінює інші налаштування і зміни повинні йти після цієї опції. Створіть типовий файл конфігурації **vim** виконуючи наступне:

```
cat > /etc/vimrc << "EOF"
" Begin /etc/vimrc
set nocompatible
set backspace=2
syntax on
if (&term == "iterm") || (&term == "putty")
set background=dark
endif
" End /etc/vimrc
EOF
```

Опція `set nocompatible` робить поведінку vim більш корисною (за звичай) ніж сумісною з vi. Видаліть “no” для отримання старої поведінки vi. Опція `set backspace=2` дозволяє видаляти розділювачі рядків, автовідступи і початки вставки. Параметр `syntax on` вмикає підсвічування синтаксису. І на кінець, вираження if разом з опцією `set background=dark` корегує здогадки **vim** про колір фону деяких емуляторів терміналів. Це дає підсвічуванню більш кращу кольорову схему для використання цих програм на чорних фонах.

Документація усіх інших доступних опцій можу бути отриманою виконанням наступної команди:

```
vim -c ':options'
```

Увага

Зазвичай, vim встановлює тільки файли spell для Англійської мови. Для встановлення файлів spell для мови, якій ви надаєте перевагу, завантажте файли `*.spl` і, опціонально, `*.sug` для вашої мови і системи кодування символів з *ftp://ftp.vim.org/pub/vim/runtime/spell/* і збережіть їх у `/usr/share/vim/vim73/spell`

Для використання цих файлів, необхідна деяка конфігурація у /etc/vimrc:

```
set spelllang=en,ru
set spell
```

Для більшої інформації, перегляньте відповідні файли README розміщені за адресою, розміщеною вище.

6.62.3. Вміст Vim

Встановлені програми: ex (посилання до vim), rview (посилання до vim), rvim (посилання до vim), vi (посилання до vim), view (посилання до vim), vim, vimdiff (посилання до vim), vimtutor і xxd

Встановлені директорії: /usr/share/vim

Короткий опис

ex Запускає vim у режимі ex

rview Обмежена версія view; ніяких команд інтерпретатора і неможливість тимчасового припинення роботи

rvim Обмежена версія vim; ніяких команд інтерпретатора і неможливість тимчасового припинення роботи

vi Посилання до vim

view Запускає vim у режимі тільки для читання

vim Редактор

vimdiff Редагує дві чи три версії файлу з vim і показує різницю

vimtutor Навчає базові комбінації клавіш і команди vim

xxd Створює шістнадцятковий дамп вказаного файлу; він також може робити реверс, отож він може бути використаний для бінарного патчування

6.63. Про символи відлагодження

Більшість програм і бібліотек є, за умовчанням, скомпільованими з символами відлагодження (з опцією компілятора gcc -g). Це означає, що про відлагоджені програми чи бібліотеки, яка скомпільована з відлагоджувальною інформацією, відлагоджувач може забезпечувати не тільки адреси пам'яті, але також імена підпрограм і змінних.

Однак, включення цих відлагоджувальних символів істотно збільшує розмір програми чи бібліотеки. Наступний список показує сумму простору які вони займають:

- бінарний файл **bash** з відлагоджувальною інформацією: 1200 Кбайт
- бінарний файл **bash** без відлагоджувальних символів: 480 Кбайт
- файли Glibc і GCC (`/lib` і `/usr/lib`) з відлагоджувальними символами: 87 Мбайт
- файли Glibc і GCC без відлагоджувальних символів: 16 Мбайт

Розміри можуть варіюватися в залежності який компілятор і бібліотека C використовувалася, але при порівнянні програм з і без відлагоджувальних символів, ці різниці можуть бути фактор між двома і п'ятьма.

Через те, що більшість користувачів ніколи не будуть використовувати символи відлагодження на їхньому

системному програмному забезпеченні, багато дискового простору можу бути відновленим видаленням цих символів. Наступна секція показує як видалити усі відлагоджувальні символи з програм і бібліотек.

6.64. Видалення відлагоджувальних символів

Якщо кінцевий користувач не є програмістом і не планує відлагодження системного програмного забезпечення, розмір системи може бути зменшений завдяки видаленню 90 Мбайт відлагоджувальних символів і бібліотек. Це спричинить неможливість відлагодження програм.

Більшість людей, які використовують команди наведені нижче, не отримують будь-які труднощі. Однак, дуже легко зробити помилку в наборі команди і спричинити несправність нової системи; отож, перед тим як виконати команду **strip**, буде хорошою ідеєю зробити резервне копіювання системи LFS у її теперішньому стані.

Перед виконанням видалення відлагоджувальних символів, попіклуйтеся про те, щоб усі бінарні файли, які підлягають опрацюванню, не були запущеними на виконання. Якщо ви не впевнені чи користувач увійшов у середовище chroot за допомогою команди даної у Секції 6.4, “Входження в середовище chroot”, для початку вийдіть з цього середовища:

```
logout
```

Після цього, зайдіть знову в нього за допомогою:

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

Тепер бінарні файли і бібліотеки можуть спокійно бути обробленими:

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

Буде повідомлено велика кількість файлів, формати яких були не розпізнаними. Ці попередження можуть бути проігнорованими. Вони показують, що ці файли є скриптами, а не бінарними файлами.

6.65. Очищення

Відтепер, після виходу з середовища chroot і в наступному входження у нього, використовуйте наступну модифіковану команду chroot:

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

Причина цього криється у тому, що ми не потребуємо програм, які знаходяться у директорії /tools. Так як вони більше нам не потрібні, вони, за бажанням, можуть бути видаленими.

> Увага
>
> Видалення `/tools` спричинить видалення тимчасових копій Tcl, Expect і DejaGNU, які використовувалися для виконання тестів. Якщо ці програми будуть потрібні вам пізніше, їх буде необхідно скомпілювати знову і перевстановити. Книга BLFS має для цього інструкції (дивіться *http://www.linuxfromscratch.org/blfs/*).

Якщо віртуальні файлові системи ядра були відключеними вручну чи через перезапуск системи, впевніться,

що віртуальні файлові системи ядра є підключеними під час входження в середовище chroot. Цей процес був роз'яснени у Секції 6.2.2, “Монтування і заповнення /dev” і секції 6.2.3, “Монтування віртуальних файлових систем ядра”.

Частина 7. Встановлення системних завантажувальних скриптів

7.1 Вступ

Ця частина обговорює конфігураційні файли і скрипти завантаження. Для початку, представляються загальні конфігураційні файли для організації мереж.

- Секція 7.2, “Загальна конфігурація мережі”.
- Секція 7.3, “Налаштовування файлу /etc/hosts”.

Наступним ділом, обговорюються питання для отримання належного встановлення пристроїв.

- Секція 7.4, “Оброблення пристроїв і модулів на системі LFS”.
- Секція 7.5, “Створення налаштованих посилань на пристрої”.

Наступна секція детально роз'яснює як встановити і налаштувати скрипти системи LFS, які необхідні для процесу завантаження. Більшість цих скриптів будуть працювати без модифікування, але деякі з них вимагають додаткових конфігураційних файлів через те, що вони мають справу з апаратно-залежною інформацією.

Ініціалізаційні скрипти, у стилі System-V, залучені у цій книзі через їхнє широке застосування і відносну простоту. Підказка, яка роз'яснює встановлення скриптів конфігурації у стилі BSD, доступна за адресою *http://www.linuxfromscratch.org/hints/downloads/files/bsd-init.txt*. Пошук списків розсилання електронних листів LFS для “depinit”, “upstart” чи “systemd” також дасть додаткову інформацію.

При використанні альтернативних скриптів ініціалізування, пропустіть дану секцію.

Список скриптів завантаження знаходяться у Апендиксі Г

- Секція 7.6, “LFS-Bootsreipts-20120901”.
- Секція 7.7, “Як ці скрипти працють?”.
- Секція 7.8, “Налаштування імя системи”.
- Секція 7.9, “Налаштування скрипту setclock”.
- Секція 7.10, “Конфігурування консолі Лінукс”.
- Секція 7.11, “Налаштування скрипту sysklogd”.

На кінець, є лаконічний вступ до скриптів і конфігураційних файлів, які використовуються при входженні користувача в систему.

- Секція 7.13, “Початкові файли оболонки Bash”.
- Секція 7.14, “Створення файлу /etc/inputrc”.

7.2 Загальна конфігурація мережі.

Дана секція застосовується тільки у випадку налаштування мережної карти.

Якщо мережна карта не буде використовуватися, немає потреби в створенні яких-небудь конфігураційних файлів для неї Якщо це ваш випадок, вам буде необхідно видалити посилання network з усіх каталогів (/etc/rc.d/rc*/d) після встановлення скриптів завантаження у Секції 7.6, “LFS-Bootscripts-20120901”.

7.2.1. Створення стабільних імен для мережних інтерфейсів

Якщо у системі присутній тільки один мережний інтерфейс, який необхідно налаштувати, не буде помилкою якщо ви виконаєте цю опціональну секцію. В багатьох випадках (наприклад, портативний комп'ютер з безпровідним і дротяним інтерфейсом), виконання вказівок налаштування цієї секції є необхідним.

З Udev і модульними мережними пристроями, нумерування мережних інтерфейсів не стійке між перезавантаженнями за замовчування через те, що драйвери завантажуються паралельно, і таким чином, у випадковому порядку. Для прикладу, на комп'ютерах, які мають два мережних карти, виготовленими Intel і Realtek, мережна карта виготовлена Intel може назватись eth0, а карта Realtek дістає ім'я eth1. У деяких випадках, після перезавантаження ці карти стають нумерованими іншим шляхом. Щоб запобігти дане явище, Udev постачається з скриптом і деякими правилами для прив'язки стабільних імен до мережних карт, які засновані на використанні MAC-адреси карти.

Ці правила були згенерованими у інструкціях побудови для udev (systemd) у останній частині. Перегляньте файл `/etc/udev/rules.d/70-persistent-net.rules`, щоб взнати які імена були прив'язані до мережних пристроїв.

```
cat /etc/udev/rules.d/70-persistent-net.rules
```

> Увага
>
> У випадках, коли MAC-адреси були прив'язаними до мережної карти вручну чи на віртуальному середовищі, такому як Xen, файл мережних правил може не бути згенерованим через те, що адреси не були послідовно прив'язаними. У таких випадках, просто продовжуйте діяти до наступної секції.

Файл починається з блоку коментарів перед двома рядками, кожен для одного NIC. Перший рядок для кожного NIC є закоментованим роз'ясненням, яке показує ID апаратного забезпечення (наприклад, постачальник PCI і ID пристрою, якщо це карта PCI), з його драйвером у дужках, якщо відповідний знайдено. Жоден ID пристрою і жоден з драйверів не використовується для визначення ім'я для даного інтерфейсу; ця інформація являється тільки довідковою. Другий рядок являється правилом Udev, яка утворює пару цього NIC і фактично прив'язує ім'я.

Усі правила Udev зроблені з декількох ключів, відокремленими комами і опціональними пробілами. Ці ключі правил і їхнє роз'яснення слідує наступним:

- `SUBSYSTEM==”net”` - Це вказує Udev ігнорувати усі пристрої, які не є мережними картами.
- `ACTION==”add”` - Це вказує Udev ігнорувати такі правила для uevent, які не є добавляючими (uevent “remove” і “change” також трапляються, але не потрбні для перейменування мережних інтерфейсів).
- `DRIVERS==”?*”` - Це існує для того, щоб Udev ігнорував мостові субінтерфейси VLAN (тому, що ці субінтерфейси не мають драйверів). Ці субінтерфейси пропускаються через те, що імена, які їм присвоюються, будуть стикатися з їхніми батьківськими драйверами.
- `ATTR{type}==”1”` - Це запевняє правило у співпаданні з головним інтерфейсом у випадку з певними драйверами безпровідних пристроїв, які створюють декілька віртуальних інтерфейсів. Інші інтерфейси пропускаються з цією самою причиною подібною з VLAN і мостовими субінтерфейсами: інакше буде зіткнення імен.
- `KERNEL==”eth*”` - Цей ключ був доданий до генератора правил Udev для обробки машин, які мають декілька мережних інтерфейсів, які мають однакові MAC-адреси (PS3 є однією з таких машин).

Якщо незалежні інтерфейси відрізняються базовими іменами, цей ключ дозволить Udev відрізняти їх один від одного. В загальному, ці дії не обов'язкові для більшості користувачів Лінукс з Початків, але вони і не зашкодять.

- NAME - Значення цього ключа є ім'я, яке Udev прив'яже до інтерфейсу.

Значення NAME є важливою частиною. Перед продовженням, впевніться, що ви знаєте які імена були прив'язані до кожної мережної карти і перевірте використання даних імен вказаних у NAME під час створення ваших конфігураційних файлів.

7.2.2. Створення конфігураційних файлів мережних інтерфейсів

Інтерфейс, який вимикається чи піднімається мережним скриптом, залежить від файлів у директорії /etc/sysconfig/. Ця директорія повинна містити файл для кожного інтерфейсу, який необхідно налаштувати, такі як ifconfig.xyz, де “xyz” - значення, яке асоціюється з інтерфейсом, наприклад, ім'я пристрою (eth0). Всередині цих файлів розміщені атрибути до цих інтерфейсів, такі як адреси IP, маски мереж і так далі. Необхідно, щоб основою ім'я цих файлів було ifconfig.

Наступні команди створюють файл-зразок для пристрою eth0 з статичною адресою IP:

```
cd /etc/sysconfig/
cat > ifconfig.eth0 << "EOF"
ONBOOT=yes
IFACE=eth0
SERVICE=ipv4-static
IP=192.168.1.1
GATEWAY=192.168.1.2
PREFIX=24
BROADCAST=192.168.1.255
EOF
```

Значення цих змінних мають бути змінені у кожному файлі для встановлення належних налаштувань.

Якщо змінна ONBOOT встановлена у значення “yes” мережний скрипт підніматиме NIC (Network Interface Card — карта мережного інтерфейсу) під час завантаження інтерфейсу. Якщо він встановлений у будь-що, крім значення “yes”, даний NIC буде проігноровано мережним скриптом, який не підніме (запустить) інтерфейс автоматично. Інтерфейс може запускатися вручну чи призупиненим за допомогою команд **ifup** і **ifdown**.

Змінна IFACE визначає ім'я інтерфейсу, для прикладу, eth0. Вона вимагається для усіх конфігураційних файлів мережних інтерфейсів.

Змінна SERVICE визначає метод, який використовується для визначення адреси IP. Пакет LFS-Bootscripts має модульний формат присвоєння IP і створення додаткових файлів у каталозі /lib/services/ дає доступ до інших методів присвоєння IP. Це використовується до розповсюдженої системи DHCP (Dynamic Host Configuration Protocol — протокл динамічного конфігурування хосту).

Змінна GATEWAY повинна містити типову адресу шлюзу, якщо такий присутній. Якщо ні, тоді закоментуйте рядок з даною змінною.

Змінна PREFIX містить число бітів, які використовуються у підмережі. Кожен октет у адресі IP має 8 бітів. Якщо маска підмережі є 255.255.255.0, тоді вона використовує перші три октети (24 біти) для специфікації номеру мережі. Якщо маска мережі є 255.255.255.240, Вона уде використовувати перші 28 біти. Префікси, довші ніж 24 біти, широко використовуються у DSL і іншими провідними ISP (Internet Service Providers). У

цьому прикладі (PREFIX=24), мережна маска є 255.255.255.0. Налаштуйте значення змінної PREFIX у відповідності до вашої специфікації мережі. Якщо даний ключ не вказано, використовується типове значення яке дорівнює 24.

Для більшої інформації перегляньте сторінку документації **ifup**.

7.2.3. Створення файлу /etc/resolv.conf

Якщо система буде підключатися до інтернету, вона потребуватиме деякі засоби перетворення імен системи DNS (Domain Name Service — сервіс доменних імен), для отримання IP адреми з доменного ім'я і навпаки. Найкраще це отримується розміщенням IP адреси серверу DNS, доступної від ISP чи мереного адміністратора, у файл `/etc/resolv.conf`. Створіть файл застосовуючи наступні команди:

```
cat > /etc/resolv.conf << "EOF"
# Begin /etc/resolv.conf
domain <Ваше доменне ім'я>
nameserver <IP адреса вашого головного сервера імен>
nameserver <IP адреса вашого допоміного сервера імен>
# End /etc/resolv.conf
EOF
```

Вираз `domain` може бути пропущеним чи заміненим виразом `search`. Дивіться сторінки документації для `resolv.conf`, щоб отримати докладнішу інформацію.

Замініть <IP адреса сервера імен> IP адресами серверів DNS, які найбільш підходять для вашого встановлення. Часто в цьому файлі буде більше ніж одне входження вираження nameserver (правила вимагають допоміжні сервери для запасних каналів отримання інформації). Якщо вам потрібно чи ви хочете вказати тільки один DNS сервер, видаліть чи закоментуйте другий вираз nameserver у конфігураційному файлі. Під IP адресою може приховуватися роутер в локальній мережі.

> Увага
>
> Публічні IPv4 адреси DNS серверів google є 8.8.8.8 і 8.8.4.4.

7.3 Налаштовування файлу /etc/hosts

Якщо будуть налаштовуватись мережні карти, визначтеся з адресами IP, FQDN (fully-qualified domain name — повне доменне ім'я), і можливі псевдоніми для використання у файлі `/etc/hosts`. Синтаксис такий:

```
[IP адреса] myhost.example.org [псевдоніми]
```

Якщо комп'ютер не буде видимим у мережі Інтернет (наприклад, існує зареєстрований домен і дійсний блок прив'язаних IP адрес — більшість користувачів цього не мають), впевніться, що IP адреса попадає в ранг приватних IP адрес мережі. Доступні ранги є:

```
Ранг приватних мережних адрес    Нормальний префікс

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
```

де, x — може бути числом у проміжку між 16-31; y може бути будь-яким числом з проміжку 0-255.

Дійові приватні IP адреса може бути 192.168.1.1. Дійовим FQDN (повне доменне ім'я) може бути lfs.example.org.

При не використанні мережної карти, все-одно вимагається дійове FQDN. Це є необхідним для коректної роботи деяких програм.

Створіть файл /etc/hosts виконуючи команди:

```
cat > /etc/hosts << "EOF"
# Початок /etc/hosts (версія мережної карти)
127.0.0.1 localhost
<192.168.1.1> <HOSTNAME.example.org> [псевдонім1] [псевдонім2 ...]
# Кінець /etc/hosts (версія мережної карти)
EOF
```

Значення <192.168.1.1> і <HOSTNAME.example.org> потребують заміни на вибрані вами чи на ті, що вимагаються (у випадку прив'язування IP адреси мережним/системним адміністратором і машина буде під'єднуватися до існуючої мережі). Опціональні псевдоніми є необов'язковими і можуть бути пропущеними.

Якщо мережна карта буде налаштовуватись, створіть файл /etc/hosts, виконуючи наступну команду:

```
cat > /etc/hosts << "EOF"
# Початок /etc/hosts (без версії мережної карти)
127.0.0.1 <HOSTNAME.example.org> <HOSTNAME> localhost
# Кінець /etc/hosts (без версії мережної карти)
EOF
```

7.4 Обробка пристроїв і модулів на системах LFS

У Частині 6, ми встановили пакет Udev. Перед тим як ми вдаватимемося в деталі його роботи, переглянемо коротку історію попередніх методів обробки пристроїв, яка наведена нижче.

Системи Лінукс, в загальному випадку, використовують статичний метод створення файлів пристроїв, за допомогою чого велика кількість файлів пристроїв створених у директорії /dev (іноді буквально тисячі файлів пристроїв), незважаючи на те чи взагалі існує відповідний апаратний пристрій. Типово це робиться скриптом **MAKEDEV**, який вміщує певну кількість викликів програми **mknod** з відповідними старшими і молодшими номерами пристроїв для кожного можливого пристрою, кий може існувати у світі.

Використовуючи метод Udev, тільки ті пристрої, які є знайденими ядром, отримують свій файл пристрою. Через те, що ці файли пристроїв будуть створеними кожен раз при завантаженні системи, вони будуть зберігатися на файловій системі devtmpfs (віртуальна файлова система, яка розміщується прямо в системній пам'яті). Файли пристроїв не вимагають багато місця, отож пам'ять яку вони займають є незначною.

7.4.1 Історія

В лютому 2000 року, нова файлова система, названа devfs, була об'єднана з ядром 2.3.46 і була доступною під час стабільних ядер серій 2.4. Хоча вона була присутньою у дереві коду, цей метод динамічного створення пристроїв ніколи не отримував достатньої підтримки від розробників ядра.

Основною проблемою з підходом прийнятим у devfs був шлях, яким він обробляє знайдення, створення і іменування пристрою. Іменування файлу пристрою було можливо найбільш критичним. Загально прийнято, що, при можливості конфігурування імен пристроїв, іменування пристроїв покладено на адміністраторів систем, а не на певних розробників. Файлова система devfs також страждає від умов конкурування, які виникають через її дизайн і не можуть бути виправленими без істотної переробки ядра. Вона була помічена

як застаріла на довгий період через відсутність обслуговування, і була остаточно видаленою з ядер у червні 2006 року.

З розробленням нестабільного ядра 2.5, пізніше випущеного як серії 2.6 стабільних ядер, прийшла нова віртуальна файлова система, названа `sysfs`. Завдання `sysfs` було у експорті поля зору конфігурації системного апаратного забезпечення до процесів простору користувача. З цим представленням у просторі користувача, можливість заміни для `devfs` стала більш реалістичною.

7.4.2. Реалізація Udev

7.4.2.1. Sysfs

Про файлову систему `sysfs` було коротко згадано раніше. Дехто може здивуватися, як sysfs знає про присутні пристрої на файловій системі і як номера пристроїв повинні використовуватись до них. Драйвери, які були скомпільованими прямо у ядро, реєструють їхні об'єкти за допомогою sysfs, при виявленні їх ядром. Для драйверів, скомпільованих як модулі, ця реєстрація настає при завантаженні модуля. Як тільки файлова система `sysfs` є монтована (у `/sys`), дані які зареєстровані вбудованими драйверами через sysfs є доступними для програм простору користувача і до **udevd** для обробки (включаючи модифікації файлів пристроїв).

7.4.2.2. Скрипти завантаження Udevd

Ініціалізуючі скрипти **/etc/rc.d/init.d/udev** піклуються за створення файлів пристроїв коли ядро Лінукс завантажене. Це робиться через те, що ядро більше не потребує запускати зовнішні бінарні файли. Замість того, **udevd** буде прослуховувати мережний сокет на предмет uevents, що генерує ядро. Це є необхідним через те, що деякі драйвери, директорії і посилання є необхідними перед доступністю динамічної обробки процесів під час ранніх стадій завантаження системи, чи потрібні самій **udevd**. Створення статичних файлів пристроїв у `/lib/udev/devices` також забезпечує робочий простір для пристроїв, які не підтримуються інфраструктурою динамічного обробки пристроїв. Скрипти завантаження запускають демон Udev, udevd, який буде діяти при отриманні будь-якої uevent. На кінець, скрипти завантаження змушують ядро згенерувати знову uevent для будь-яких пристроїв, які вже зареєстровані і після цього чекає щоб **udevd** обробив їх.

Скрипти ініціалізування **/etc/rc.d/init.d/udev_retry** піклуються про повторне генерування uevents для підсистем, правила яких можуть залежати на файлових системах, які не були змонтованими поки не виконуються скрипти mountfs (злкрема, `/usr` і `/var` можуть спричинити це). Цей скрипт виконуються після скрипта mountfs, отож ці правила (після повторної генерації) повинні виконатись успішно наступного разу. Вони налаштовуються у файлі `/etc/sysconfig/udev_retry`; будь-які слова у цьому файлі, відмінні від коментарів, вважаються іменами підсистем для запуску при повторній спробі. (Для знаходженні підсистем пристрою, використовуйте **udevadn info --attribute-walk**).

7.4.2.3. Створення файлів пристроїв

В останніх версіях udev, **udevd** більше не створює файли пристроїв у `/dev`. Замість того, це має бути виконане ядром, файловою системою `devtmpfs`. Будь-який драйвер, який “бажає” зареєструвати файл пристрою, буде це робити через `devtmpfs` (ядром драйверу). Коли `devtmpfs` є змонтованою на `/dev`, файл пристрою спочатку буде створений з фіксованим ім'ям, правами доступу і власником.

Через короткий момент, ядро пошле uevent до **udevd**. На основі правил специфікованих у файлах в директоріях `/etc/udev/rules.d`, `/lib/udev/rules.d`, і `/run/udev/rules.d`, **udevd** створить

додаткові посилання до файлів пристроїв, чи змінить його права доступу, власника чи групи, або модифікує записи внутрішньої бази даних **udevd** для цього об'єкту.

Правила у цих трьох директоріях є пронумерованими у подібній манері до пакету LFS-Bootscripts, і усі три директорії є об'єднані разом. Якщо **udevd** не може знайти правила для пристроїв які вона створює, вона залишить права доступу і власника такими, якими їх зробила `devtmpfs`.

7.4.2.4 Завантаження модулів

файли пристроїв скомпільовані як модулі можуть мати вбудовані в них псевдоніми. Псевдоніми є видимими у виводі програми **modinfo** і зазвичай є пов'язані з ідентифікаторами у шині для пристроїв, які підтримуються модулем. Для прикладу, драйвери `snd-fm801` підтримує пристрої PCI з ID розробника 0x1319 і з ID пристрою 0x0801, і має псевдонім “pci:v00001319d00000801sv*sd*bc04sc01i* ”. Для більшості пристроїв, драйвер шини експортує псевдонім драйверу, який буде обробляти пристрій sysfs. Наприклад, файл `/sys/bus/pci/devices/0000:00:0d.0/modalias` може містити рядок “pci:v00001319d00000801sv00001319sd00001319bc04sc01i00”. Типові правила як постачаються з Udev будуть спричиняти **udevd** викликати **/sbin/modprobe** з вмістом uevent змінної середовища `MODALIAS` (яка повинна бути такою ж як і вміст файлу modalias у sysfs), та завантажує усі модулі, псевдоніми яких збігаються з цим рядком після розкриття шаблонів.

У цьому прикладі, спосіб який, у додаток до `snd-fm801`, застарілий (і непотрібний) драйвер forte буде завантаженим, якщо він є доступним. Дивіться нижче для шляхів які запобігають завантаження непотрібних драйверів.

Саме ядро також може завантажувати модулі для мережних протоколів, файлових систем і підтримки NLS за потреби.

7.4.2.5. Обробка динамічних пристроїв

Коли ви підключаєте пристрій, такі як USB (Universal Serial Bus — універсальна послідовна шина) плеєр MP3, ядро визначеє, що підключено пристрій і генерує uevent. Ця uevent обробляється **udevd** як було показано вище.

7.4.3. Проблеми з завантаженням модулів і створенням пристроїв

Є декілька можливих проблем, які зв'язані з автоматичним створенням файлів пристроїв.

7.4.3.1 Модуль ядра не завантажено автоматично

Udev завантажить модуль тільки якщо вона має псевдонім специфікований шиною і драйвер шини належно експортує необхідні псевдоніми до sysfs. У інших випадках, вона повинна налаштувати завантаження модулів іншим шляхом. З Linux-3.5.2, відомо, що Udev належно завантажує модулі для драйверів INPUT, IDE, PCI, USB, SCSI, SERIO і FireWire.

Щоб визначити чи підтримується драйвер необхідного пристрою для Udev, виконайте **modinfo** з ім'ям модуля як аргумент. Тепер спробуйте визначити директорію пристрою у `/sys/bus` і перевірте чи там є файл `modalias`.

Якщо файл `modalias` існує у `sysfs`, драйвер підтримує пристрій і може спілкуватися з ним прямо, але не має псевдонім, це означає що є помилка у драйвері. Завантажуйте драйвер без допомоги від Udev і чекайте на вирішення проблеми пізніше.

Якщо там немає ніякого файлу `modalias` у відповідній директорії в `/sys/bus`, це означає, що розробники ядра ще не додали підтримку modalias до цього типу шини. З Linux-3.5.2, це є випадок з шинами ISA. Очікуйте вирішення цих питань у пізніших версіях.

Udev зовсім не призначена для завантаження драйверів-обгорток, таких як snd-pcm-oss і драйверів не для пристроїв такиз як `loop`.

7.4.3.2. Модуль ядра не завантажений автоматично, і Udev не пристосована для його завантаження.

Якщо модуль-обгортка підвищує функціональність забезпечену якимось іншими модулями (наприклад, snd-pcm-oss підвищує функціональність snd-pcm, роблячи звукові карти доступними для програм OSS), налаштуйте modprobe на завантаження обгортки після того, як Udev завантажить модуль. Щоб зробити це, додайте рядок “softdep” в будь-який файл `/etc/mdprobe.d/<ім'я файлу>.conf`. Для прикладу:

```
sodtdep snd-pcm post: snd-pcm-oss
```

Зверніть увагу на те, щл команда “softdep” також дозволяє залежності pre:, чи суміш обох pre: і post:. Дивіться сторінку документації modprobe.d(5) для більш детального роз'яснення синтаксису “softdep” і її можливостей.

Якщо модуль не є обгорткою і є корисним, налаштуйте скрипти завантаження modules на завантаження цього модуля під час завантаження системи. Щоб зробити це, долайте ім'я модуля до файлу /etc/sysconfig/modules у окремому рядку. Це також працює і для модулів-обгорток, але не є оптимально у цьому випадку.

7.4.3.3. Udev завантажує деякі непотрібні модулі

Або не будуйте модуль, чи занесіть його у файл `/etc/modprobe.d/blacklist.conf` як це було зроблено з модулем *`forte`* у прикладі наведеному нижче:

```
blacklist forte
```

Модулі занесені у чорний список можуть все одно бути завантажені вручну за допомогою програми modprobe.

7.4.3.4 Udev створює пристрої некоректно, чи робить неправильні посилання

Це зазвичай трапляється, якщо правило неочікувано подібне до пристрою. Для прикладу, погано написане правило можу збігатися з диском SCSI і відповідним загальним пристроєм SCSI за виробником. Знайдіть некоректне правило і зробіть його більш точним за допомогою допомоги команди **udevadm info.**

7.4.3.5. Правила Udev працюють ненадійно

Це може бути іншим проявом попередньої проблеми. Якщо ні, і ваші правила використовують атрибути *`sysfs`*, це може бути проблемою часу ядра, яке буде виправлено у пізніших ядрах. Зараз, ви можете працювати з цим створюючи правила, які чекають на виконання атрибуту sysfs і додавання його до файлу `/etc/udev/rules.d/10-wait_for_sysfs.rules` (створіть цей файл, якщо він не існує). Будь-ласка, повідомте список розроблення LFS якщо ви зробили це і воно допомогло.

7.4.3.6. Udev не створює пристрій

Подальший текст припускає, що драйвер вбудований статично у ядро чи вже завантажений як модуль, і ви вже перевірили, що Udev не створює невірно названий пристрій.

Udev не має ніякої інформації потрібної для створення файлів пристроїв, якщо драйвер ядра не експортує його дані до `sysfs`. Це є найбільш поширеною проблемою з не критичними драйверами ззовні дерева ядра. Створіть статичний файл пристрою у директорії `/lib/udev/devices` з належними старшими і молодшими числами (дивіться файл `devices.txt` всередині документації ядра чи документації забезпеченої стороннім виробником драйверу). Статичний пристрій буде скопійований у каталог `/dev` скриптом завантажерня udev.

7.4.3.7. Порядок іменування пристроїв випадково змінюється після перезавантаження

Це пов'язано з тим що Udev, як і розроблялось, обробляє uevents і завантажує модулі паралельно, і це відбувається в непередбачуваному порядку. Це ніколи не буде “виправлено”. Ви повинні не покладатися на те, що імена пристроїв ядра будуть стабільними. Замість цього, створіть ваші власні правила, які зроблять посилання на стабільні імена, засновані на деяких атрибутах пристроїв, таких як серійні номера чи вивід деяких утиліт *_id замість Udev. Перегляньте Секцію 7.5, “Створення користувацьких посилань на пристрої” і Секцію 7.2, “Залальна конфігурація мережі” для прикладу.

7.4.4. Корисна інформація

Додаткова корисна документація доступна з наступних сторінок інтернету:

- Реалізація програм простору користувача `devfs` *http://www.kroah.com/linux/talks/ols_2003_udev_paper/Reprint-Kroah-Hartman-OLS2003.pdf*
- Файлова система sysfs http://www.kernel.org/pub/linux/kernel/people/mochel/doc/papers/ols-2005/mochel.pdf
- Вказівки для подальшого читання http://www.kernel.org/pub/linux/utils/kernel/hotplug/udev.html

7.5 Створення користувацьких посилань на пристрої

7.5.1 Посилання пристрою CD-ROM

Деякі програми, що ви можете забажати встановити пізніше (наприклад, декілька медіа плеєрів) очікують існування посилань `/dev/cdrom` і `/dev/dvd`, які вказують на пристрої CD-ROM чи DVD-ROM. Також, це можу бути зручним для вказування вказівки про ці посилання у `/etc/fstab`. Udev постачається з скриптом, який згенерує файли правил для створення цих посилань для вас, в залежності від можливостей кожного пристрою, але вам необхідно вирішити який з двох режимів операцій ви бажаєте щоб скрипт використовував.

По-перше, скрипт може оперувати у режимів “за-шляхом” (типово використовується для пристроїв USB і FireWire), де правила які він використовує залежать від фізичного шляху до пристроїв CD чи DVD. По-друге, він може оперувати у режимі “за-ідентифікатором” (типовий для пристроїв IDE і SCSI), де правила за якими він створює файли пристроїв залежить від ідентифікаторів-рядків, які збережені у самих пристроях CD чи DVD. Шлях визначається скриптом Udev — **path_id**, і рядки-ідентифікатори читаються з пристроїв за допомогою програмам **ata_id** чи **scsi_id**, в залежності який тип пристрою ви маєте.

Переваги мають кожний з підході; використання коректного підходу залежить від того, які типи змін пристроїв можуть статися. Якщо ви очікуєте зміну фізичного шляху до пристроїв (себто, порти і/чи слоти в які вони увімкнені), тоді ви повинні використовувати режим “за-ідентифікаторм”. З іншого боку, якщо ви очікуєте зміну ідентифікаторів пристроїв, для прикладу через те що він може поламатись і ви замінiть його

іншим пристроєм з такими ж можливостями і які під'єднуються у ті ж конектори, в такому випадку ви повинні використовувати режим “за-шляхом”.

Якщо можливий обидві зміни у конфігурації пристроїв, тоді оберіть режим, який заснований на типі зміни, яку ви очікуєте найбільш часто.

> Важливо
>
> Зовнішні пристрої (для прикладу, CD-привід підключений через USB) не повинні використовувати режим за-шляхом через те, що кожного разу, коли пристрій підключений у новий зовнішній порт, його фізичний шлях зміниться. Усі підключенні ззовні пристрої будуть мати цю проблему, якщо ви напишете правила Udev розпізнавати їх за їхнім фізичним шляхом; проблема не обмежується для пристроїв CD чи DVD.

Якщо ви бажаєте переглянути значення, які буде використовувати скрипт Udev, для відповідного пристрою CD-ROM, знайдіть відповідну директорію під `/sys` (наприклад, це може бути `/sys/block/hdd`) і виконайте команду подібну до наступної:

```
udevadm test /sys/block/hdd
```

Перегляньте рядки які містять вивід деяких програм *_id. Режим “за-ідентифікатором” буде використовувати значення ID_SERIAL якщо він існує і не пустий, в іншому випадку він буде використовувати комбінацію ID_MODEL і ID_REVISION. Режим “за-шляхом” буде використовувати значення ID_PATH.

Якщо типовий режим не підходить до вашої ситуації, тоді наступні модифікації можуть бути зробленими до файлу `/lib/udev/rules.d/75-cd-aliases-generator.rules` (де `mode` є одним з “за-ідентифікаторм” (“by-id”) чи “за-шляхом” (“by-path”)):

```
sed —i -e 's/"write_cd_rules"/"write_cd_rules mode"/' \
/lib/udev/rules.d/75-cd-aliases-generator.rules
```

Зверніть увагу на те, що це не необхідно створювати файли правил чи посилання цього разу через те, що ви маєте зв'язану монтовану хостову директорію `/dev` у системі LFS і ми припускаємо, що посилання існують на хості. Правила і посилання будуть створені при першому запуску системи LFS.

Однак, якщо ви маєте декілька пристроїв CD-ROM, тоді посилання згенеровані в цей час можуть вказувати на інші пристрої ніж під час того як вони вказували на вашому хості через те, що пристрої не були знайденими у передбачуваному порядку. Завдання створені під час вашого першого завантаження системи LFS будуть стабільними, отож це є тільки проблемою, якщо вам потрібні посилання на обох системах для вказування того самого пристрою. Якщо це вам необхідно, тоді перевірте (і можливо відредагуйте) згенерований файл `/etc/udev/rules.d/70-persistent-cd.rules` після завантаження, щоб впевнитися чи збігаються посилання вашим вимогам.

7.5.2 Розрахунок дубльованих пристроїв

Як було роз'яснено у Секції 7.4, “Обробка модулів і пристроїв на системах LFS”, порядок у якому пристрої з однаковими функціями з'являються у /dev є по суті випадковим. Наприклад, якщо ви маєте USB веб камеру і TV тюнер, інколи `/dev/video` посилається а камеру і `/dev/video1` посилається на тюнер, інколи після перезавантаження порядок змінюється на протилежний. Для усіх класів апаратного забезпечення замість звукових і мережних карт, це може бути виправленим створенням правил udev для стійких посилань. Випадок мережних карт роз'яснюється у Секції 7.2 “Загальна конфігурація мережі”, і конфігурація звукових карт може бути знайденою у *BLFS*.

Для кожного вашого пристрою, є шанс отримати цю проблему (навіть якщо проблема не існує у вашому поточному дистрибутиві Лінукс), знайдіть відповідну директорію під /sys/class чи /sys/block. Для відеопристроїв, це можу бути /sys/class/video4linux/videoX. Взнайте атрибути, які однозначно ідентифікують пристрій (зазвичай, ID постачальника і продукту і/чи серійні номери роботи):

```
udevadm indo -a -p /sys/class/video4linux/video0
```

Тоді напишіть правила, які створюють посилання, наприклад:

```
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
```

Результатом є те, що пристрої `/dev/video0` і `/dev/video1` досі посилаються на випадково на камеру і на тюнер (і вони не повинні прямо використовуватися), але існують посилання `/dev/tvtuner` і `/dev/webcam`, які завжди посилаються на коректний пристрій.

7.6. LFS-Bootscripts-20120901

Пакет LFS-Bootscripts містять набір скриптів для старту/зупинки системи LFS при завантаженні/вимиканні.

Приблизний час побудови: менш ніж 0.1 SBU
Необхідний дисковий простір: 260 Кбайт

7.6.1. Встановлення LFS-Bootscripts

Встановіть пакет

```
make install
```

7.6.2. Вміст LFS-Bootscripts

Встановлені скрипти: checkfs, cleanfs, console, functions, halt, ifdown, ifup, localnet, modules, mountfs, mountkernfs, network, rc, reboot, sendsignals, setclock, static, swap, sysctl, sysklogd, template, udev і udev_retry

Встановлені директорії: /etc/rc.d, /etc/init.d (symbolic link), /etc/sysconfig, /lib/services, /lib/lsb (символічне посилання)

Короткий опис

checkfs Перевіряє цілісність файлових систем перед тим як їх монтувати (за вийнятку журнальних і мережних файлових систем)

cleanfs Видаляє файли які не повинні зберегтися між перезавантаженнями, такі як у директоріях /var/run і /bar/lock; вона знову створює /var/run/utmp і видаляє можливо присутні файли /etc/nologin, /fastboot, і /forcefsck

console Завантажує коректну таблицю карти ключів для бажаної розкладки клавіатури; вона також встановлює шрифт екрану

functions Містить поширені функції, такі як помилки і перевірка статусів, які використовуються деякими скриптами завантаженнями

halt Призупиняє систему

ifdown Зупиняє мережні інтерфейси

ifup Ініціалізовує мережні пристрої

localnet Встановлює ім'я хосту і локальний дзеркальний пристрій

modules Завантажує модулі ядра які перераховані у /etc/sysconfig/modules, використовуючи аргументи які також дані в цьому файлі

mountfs Монтує віртуальні файлові системи ядра, такі як proc

network Встановлює мережні інтерфейси, такі як мережні карти, і встановлює

	типовий шлюз (де це можливо)
rc	Головний скрипт контролю рівнів виконання; він відповідальний за виконання усіх інших скриптів один-за-одним, у послідовності визначеній оброблених імен символічних посилань
reboot	Перезавантажує систему
sendsignals	Перевіряє чи усі процеси завершені перед тим як система перезавантажується чи вимикається
setclock	Збиває годинник ядра до локального часу у випадку коли апаратний годинник не встановлений у час UTC
static	Забезпечує необхідну функціональність для прив'язки статичних адрес IP до мережного інтерфейсу
swap	Вмикає чи вимикає файли і розділи swap
sysctl	Завантажує системні конфігураційні значення з /etc/sysctl.conf, якщо такі існують у запущене ядро
sysklogd	Запускає чи зупиняє демони журналів системи і ядра
template	Шаблон для створення користувацьких скриптів завантаження для демонів
udev	підготовлює директорію /dev і запускає Udev
udev_retry	Перезапускає провалені події (uevents) udev і якщо необхідно копіює файли правил з /etc/udev/rules.d

7.7. Як працюють ці скрипти завантаження

Лінукс використовує спеціальний об'єкт завантаження SysVinit, який заснований на концепції рівнів виконання. Вона може дещо відрізнятися від однієї системи на іншій, отже через те це може бути передбачено, що речі які працюють на одній системі Лінукс дистрибутива, повинні також працювати на такій самі системі LFS. LFS також має його власний шлях виконання дій, але він також поважає прийняті стандарти.

SysVinit (відтепер, на який ми будемо посилатися як “init”) працює використовуючи схему рівнів виконання. Існує сім (від 0 до 6) рівнів виконання (взагалі, існує більше рівнів виконання, але вони є спеціалізованими випадками і в загальному не використовуються; дивіться init(8) для докладнішої інформації), і кожен з цих рівнів відповідає дій комп'ютера які очікується на виконання коли він запускається. Типовий рівень виконня рівний 3. Є роз'яснення інших рівнів виконання так як вони є реалізованими:

```
0: вимкнути комп'ютер
1: однокористувацький режим
2: багатокористувацький режим без функцій мережі
3: багатокористувацький режим з функціями мережі
4: збережено для налаштування, інакше виконує те саме що і 3
5: так само як і 4, він зазвичай використовується для входу через GUI
```

```
(наприклад X xdm чи KDE kdm)
6: перезавантажимти комп'ютер
```

7.1.1. Конфігурування Sysvinit

Під час ініціалівзування ядра, першою програмою яка виконується є специфікована в командному рядку, якщо ні — типово використовується **init**. Ця програма читає ініціалізуючі файли `/etc/inittab`. Створіть цей файл за допомогою:

```
cat > /etc/inittab << "EOF"
# Begin /etc/inittab
id:3:initdefault:
si::sysinit:/etc/rc.d/init.d/rc S
l0:0:wait:/etc/rc.d/init.d/rc 0
l1:S1:wait:/etc/rc.d/init.d/rc 1
l2:2:wait:/etc/rc.d/init.d/rc 2
l3:3:wait:/etc/rc.d/init.d/rc 3
l4:4:wait:/etc/rc.d/init.d/rc 4
l5:5:wait:/etc/rc.d/init.d/rc 5
l6:6:wait:/etc/rc.d/init.d/rc 6
ca:12345:ctrlaltdel:/sbin/shutdown -t1 -a -r now
su:S016:once:/sbin/sulogin
1:2345:respawn:/sbin/agetty
2:2345:respawn:/sbin/agetty
3:2345:respawn:/sbin/agetty
4:2345:respawn:/sbin/agetty
5:2345:respawn:/sbin/agetty
6:2345:respawn:/sbin/agetty
--noclear tty1 9600
tty2 9600
tty3 9600
tty4 9600
tty5 9600
tty6 9600
# End /etc/inittab
EOF
```

Роз'яснення цього ініціалізаційного файлу є у сторінці документації для *inittab*. Для LFS, ключовою командою виконується є rc. Ініціалізаційний файл наведений вище, проінструктує rc виконати усі скрипти які починаються на S у директорії `/etc/rc.d/rcsysinit.d` які йдуть шляхом за усіма скриптами які починаються на S у каталозі `/etc/rc.d/rc?.d` де знак питання специфікується типовим значенням init.

Для зручності, скрипт rc читає бібліотеку функцій у `/lib/lsb/init-functions`. Ця бібліотека також читає опціональні конфігураційні файл, `/etc/sysconfig/rc.site`. Будь-який з системних параметрів конфігураційних файлів описаний у наступних секціях може бути альтернативно поміщений у цьому файлі, дозволяючи об'єднання усіх системних параметрів у цьому одному файлі.

Для зручності відлагодження, скрипт функцій також зберігає в журнал весь вивід до `/run/var/bootlog`. Так як директорія `/run` є `tmpfs`, цей файл не є стійким крізь перезавантаження, однак він додається до більш постійного файлу `/var/log/boot.log` після завершення процесу завантаження.

7.7.2. Зміна рівнів виконання

Змінюючи рівні виконання виконується за допомогою команди **init <рівень_виконання>**, де `<рівень_виконання>` є цільовим рівнем виконання. Для прикладу, для перезавантаження комп'ютера, користувач може виконати команду **init 6**, що є псевдонімом для команди **reboot**. Також, **init 0** є

псевдонімом для команди **halt**.

Є декілька директорій під `/etc/rc.d` які виглядають як `rc?.d` (де ? є номер рівня виконання) і `rcsysinit.d`, які вміщають певну множину символічних посилань. Деякі починаються з K, інші починаються з S, і усі вони мають два номери наступними після початкової літери. Літера K означає призупинити (kill) сервіс і S означає розпочати сервіс. Числа визначають порядок у якому скрипти виконуються, з 00 до 99 — чим менше число, ти раніше запускається скрипт. Коли init перемикається у інший рівень виконання, відповідні сервіси є запущеними чи зупиненими, в залежності від вибраного рівня виконання.

Є один винятком до даного роз'яснення. Посилання, які розпочинаються з S у директоріях `rc0.d` і `rc6.d` не спричинять ніякого запуску. Вони будуть викликатися з параметром `stop` для призупинення чогось. Логіка, яка стоїть за цим у тому, що при перезавантаженні користувачем комп'ютера чи вимкнення системи, ніщо не повинно бути запущеним. Система потребує бути зупиненою. Це є роз'яснення до яких дій призводить аргумент:

`start`

Запускає сервіс.

`stop`

Призупиняє сервіс.

`restart`

Сервіс є призупиненим і після цього знову запущеним

`reload`

Конфігурація сервісу була оновленою. Це використовується після модифікування конфігураційних файлів сервісів, коли сервіс не потребує у перезапуску.

`status`

Надає інформацію чи сервіс є запущеним і з яким PID-ом.

Ви можете вільно модифікувати шлях роботи процесу завантаження (в кінці кінців, це ж ваша система LFS). Файли дані тут є прикладом того, як це може бути зроблено.

7.8 Налаштовування системно ім'я хосту

Частиною роботи скрипту **localnet** є встановлення системного ім'я машини. Це потребує налаштовування у файлі `/etc/sysconfg/network`.

Створіть файл і введіть ім'я машини виконанням наступної команди:

```
echo "HOSTNAME=<lfs>" > /etc/sysconfig/network
```

`<lfs>` потребує бути заміненим вибраним ім'ям комп'ютера. Не вводьте тут FQDN. Ця інформація вказується у файлі `/etc/hosts`.

7.9. Конфігурація скрипту setclock

Скрипт **setclock** читає час з апаратного годинника, також відомого як годинник BIOS чи CMOS (Complementary Metal Oxide Semiconductor). Якщо годинник апаратного забезпечення встановлений у UTC, скрипт перетворить апаратний час до локального використовуючи файл /etc/localtime (який вказує програмі

hwclock в якому часововому поясі являється користувач). Немає шляху визначити чи апаратний годинник встановлено у UTC чи ні, отож це потребує ручного налаштування.

Програма **setclock** запускається udev коли ядро визначає можливості апаратури під час завантаження. Вона також може бути виконаною вручну з параметрами stop для збереження системного часу у годинник CMOS.

Якщо ви не пам'ятаєте чи встановлено апаратний годинник у UTC, визначте це виконуючи команду **hwclocl --localtime --show**. Це покаже, який поточний час встановлено у системний годинник. Якщо час співпадає з локальним часом, є шанс, що він встановлений у час UTC. Перевірте це додаванням чи відніманням належної кількості годин для часової зони до часу показаного програмою hwclock. Для прикладу, якщо ви у часовій зоні MST, що також відома як GMT -0700, добавте сім годин до локального часу.

Змініть значення змінної UTC нижче до значення 0 (нуль) якщо апаратний годинник не встановлений у час UTC.

Створіть новий файл `/etc/sysconfig/clock` виконуючи наступне:

```
cat > /etc/sysconfig/clock << "EOF"
# Begin /etc/sysconfig/clock
UTC=1
# Set this to any options you might need to give to hwclock,
# such as machine hardware clock type for Alphas.
CLOCKPARAMS=
# End /etc/sysconfig/clock
EOF
```

Хорошою підказкою, яка роз'яснює як працювати з часом на LFS є доступною за адресою *http://www.linuxfromscratch.org/hints/downloads/files/time.txt*. Вона роз'яснює питання такі як часові зони, UTC, і змінну середовища TZ.

> Увага
>
> Параметри CLOCKPARAMS і UTC можуть бути альтернативно встановленими у файлі /etc/sysconfig/rc.site.

7.10. Налаштування консолі Лінукс

Ця секція обговорює як налаштувати скрипти завантаження **console**, які встановлюють карту клавіатури, консольний шрифт і рівень запису в журнали ядра. Якщо не-ASCII символи (наприклади, знак копіювання, знак британського фунта і символ Євро) не будуть використовуватися і клавіатура являється клавіатурою U.S., більшість цієї секції можна пропустити. Без конфігураційного файлу, (чи еквівалентних налаштувань у `rc.site`), скрипт завантаження **console** не буде робити нічого.

Скрипт **console** читає файл `/etc/sysconfig/console` для конфігураційної інформації. Виберіть яку карту символів і екранний шрифт буде використовуватися. Деякі специфіковані для мов HOWTO також можуть допомогти у цьому, дивіться *http://www.tldp. org/HOWTO/HOWTO-INDEX/other-lang.html*. Якщо все-одно сумніваєтесь, подивіться у директорію `/lib/kdb` для дійових карт символів і екранних шрифтів. Читайте сторінки документації `loadkeys (1)` і `setfont (8)` для визначення коректних аргументів для цих програм.

Файл `/etc/sysconfig/console` повинен містити рядки у формі: ЗМІННА=”значення”. Розпізнаються наступні змінні:

LOGLEVEL

Ця змінна специфікує рівень журналювання для повідомлень ядра надісланих до консолі як множину dmesg. Дійові рівні мають проміжок від “1” (ніяких повідомлень) до “8”. Типовий рівень рівний “7”.

KEYMAP

Ця змінна специфікує аргументи для програм loadkeys, типово, ім'я карти символів для завантаження, наприклад “uk”. Якшо ця змінна не встановлена, скрипти завантаження не виконають програму loadkeys, і буде використовуватися типова карта символів ядра.

KEYMAP_CORRECTIONS

Ця (рідко використовувана) змінна специфікує аргументи для другого виклику програми loadkeys. Це корисно якщо стокова карта символів не повністю задовільна, і потрібно зробити деяке редагування. Наприклад, для включення знаку Євро у карту символів, яка типово його немає, встановіть цю змінну у “euro2”.

FONT

Ця змінна специфікує аргументи для програми setfont. Типово, вона включає в себе назву шрифту, “-m”, і ім'я програми символьної карту, яку необхідно завантажити. Наприклад, за порядком завантаження шрифт “lat1-16” разом з картою символів програми “8859-1” (так як вона доречна для США), встановіть цю змінну у “lat1-16 -m 8859-1”. У режимі UTF-8, ядро використовує карту символів програми для перетворення складних 8-бітних кодів символів у розкладці до UTF-8 і таким чином аргумент параметру “-m” повинен бути встановленим у кодування складних кодів символів у розкладці.

UNICODE

Встановіть цю змінну у “1”, “yes” чи “true” в порядку встановлення консолі у режим UTF-8. Це є корисним для локаль заснованих на UTF-8 і шкідливо для усіх інших.

LEGACY_CHARSET

Для багатьох розладок клавіатур, немає ніякої наявної карти символів Unicode у пакеті Kdb. Скрипт завантаження console перетворить доступну карту ключів до UTF-8 під час роботи, якщо ця змінна встановлена до кодування змінної не UTF-8 карт символів.

Деякі приклади

- Для встановлення не Unicode, потрібні тільки змінні KEYMAP і FONT. Наприклад для встановлення польської використовується:

```
cat > /etc/sysconfig/console << "EOF"
# Begin /etc/sysconfig/console
KEYMAP="pl2"
FONT="lat2a-16 -m 8859-2"
# End /etc/sysconfig/console
EOF
```

- Як було згадано вище, інколи необхідно трішки відрегулювати наявну карту символів. Наступний приклад додає символ Євро до загальної карти:

```
cat > /etc/sysconfig/console << "EOF"
# Begin /etc/sysconfig/console
KEYMAP="de-latin1"
```

```
KEYMAP_CORRECTIONS="euro2"
FONT="lat0-16 -m 8859-15"
# End /etc/sysconfig/console
EOF
```

- Наступний приклад вмикає Unicode для болгарської, де існує наявна карта ключів UTF-8:

```
cat > /etc/sysconfig/console << "EOF"
# Begin /etc/sysconfig/console
UNICODE="1"
KEYMAP="bg_bds-utf8"
FONT="LatArCyrHeb-16"
# End /etc/sysconfig/console
EOF
```

- Через використанння 512-гліфного шрифту LatArCyrHeb-16 у попередньому прикладі, яскраві кольори більше не доступні на консолі Лінукс, тільки якщо використовується фреймбуфер. Якщо є бажання мати яскраві кольори без фреймбуфера і готовність обійтись без символів які належать до даної мови, все ж можливо використовувати специфічні для мови 256-гліфні шрифти, як проілюстровано нижче:

```
cat > /etc/sysconfig/console << "EOF"
# Begin /etc/sysconfig/console
UNICODE="1"
KEYMAP="bg_bds-utf8"
FONT="cyr-sun16"
# End /etc/sysconfig/console
EOF
```

- Наступний приклад ілюструє автоперетворення карти ключів з ISO-8859-15 до UTF-8 і вмикання мертвих ключів у режимі Unicode:

```
cat > /etc/sysconfig/console << "EOF"
# Begin /etc/sysconfig/console
UNICODE="1"
KEYMAP="de-latin1"
KEYMAP_CORRECTIONS="euro2"
LEGACY_CHARSET="iso-8859-15"
FONT="LatArCyrHeb-16 -m 8859-15"
# End /etc/sysconfig/console
EOF
```

- Деякі карти мають мертві ключі (наприклад, ключі які не продукують самостійно символи, але накладають акцент на символи продуковані наступною клавішою) чи складені правила (такі як: “натисніть Ctrl+.A E для отримання Æ ” у типовій клавіатурі). Linux-3.5.2 інтерпретує мертві клавіші і складені правила у карті ключів коректно тільки тоді, коли складені разом початкові символи не є мультибайтовими. Це дефіцитно не впливає на коавіатури для європейських мов через те, що в них акценти додаються неакцентованих символів ASCII, чи два символи ASCII складені разом. Однак, у режимі UTF-8 це є проблемою, наприклад для грецької мови, де інколи необхідно поставити акцент на символ “альфа”. Вирішенням буде або заборона використання UTF-8, чи встановити віконну систему X, яка не має цього обмеження у її обробці введення.

- - Для китайської, японської, корейської і інших мов, консоль Лінукс не може бути налаштованим для виводу потрібних символів. Користувачі які потребують таких мов повинні встановити систему X Window System, шрифти які перекривають необхдні ранги символів і належний метод вводу (наприклад, SCIM, він підтримує широкий спектр мов).

Увага

Файл /etc/sysconfig/console контролює тільки текстову консольну локалізацію Лінукс. Вона не має ніякого відношення до відповідних розкладок клавіатур і термінальних шрифтів системи X Window System, з сесіями ssh чи серійними консолями. У таких випадках, обмеження згадані у останніх двох пунктах не застосовуються.

7.11 Налаштування скрипту sysklogd

Скрипт sysklog викликає програму syslogd з опціямєю -m 0. Ця опція вимикає періодичну часову мітку, яку syslogd записує до журнальних файлів типово кожні 20 хвилин. Якщо ви хочете увімкнути цю періодичну часову мітку, відредагуйте /etc/sysconfig/rc.site і визначте змінну SYSKLOGD_PARMS до бажаного значення. Для прикладу, видалити усі параметри, встановити змінну у нульове значення:

```
SYSKLOGD_PARAMS=
```

Для усіх інших опцій дивіться сторінку документації **man syslogd.**

7.12. Файл rc.site

Необов'язковий файл `/etc/sysconfig/rc.site` містить налаштування, які автоматично встановлюються для кожного скрипту завантаження. Він може встановити альтернативні значення встановлені у файлах `hostname`, `console` і `clock` у директорії `/etc/sysconfig/`. Якщо пов'язані змінні присутні у обох цих окремих фалах і `rc.site`, значення у файлах скриптів мають пріоритет.

Файл `rc.site` також вміщує параметри, які можуть налаштовувати інші аспекти процесу завантаження. Встановлюючи змінну IMPORT увімкне вибіркове виконання скриптів завантаження. Інші опції описані у коментаріях файлу. Типова версія цього файлу є наступною:

```
# rc.site
# Optional parameters for boot scripts.
# Distro Information
# These values, if specified here, override the defaults
#DISTRO="Linux From Scratch" # The distro name
#DISTRO_CONTACT="lfs-dev@linuxfromscratch.org" # Bug report address
#DISTRO_MINI="LFS" # Short name used in filenames for distro config

# Define custom colors used in messages printed to the screen

# Please consult `man console_codes` for more information
# under the "ECMA-48 Set Graphics Rendition" section
#
# Warning: when switching from a 8bit to a 9bit font,
# the linux console will reinterpret the bold (1;) to
# the top 256 glyphs of the 9bit font. This does
# not affect framebuffer consoles
#

# These values, if specified here, override the defaults
#BRACKET="\\033[1;34m" # Blue
#FAILURE="\\033[1;31m" # Red
#INFO="\\033[1;36m"
# Cyan
#NORMAL="\\033[0;39m" # Grey
```

```
#SUCCESS="\\033[1;32m" # Green
#WARNING="\\033[1;33m" # Yellow

# Use a colored prefix
# These values, if specified here, override the defaults
#BMPREFIX="    "
#SUCCESS_PREFIX="${SUCCESS} * ${NORMAL}"
#FAILURE_PREFIX="${FAILURE}*****${NORMAL}"
#WARNING_PREFIX="${WARNING} *** ${NORMAL}"

# Interactive startup
#IPROMPT="yes" # Whether to display the interactive boot prompt
#itime="3"
# The amount of time (in seconds) to display the prompt

# The total length of the distro welcome string, without escape codes
#wlen=$(echo "Welcome to ${DISTRO}" | wc -c )
#welcome_message="Welcome to ${INFO}${DISTRO}${NORMAL}"
# The total length of the interactive string, without escape codes
#ilen=$(echo "Press 'I' to enter interactive startup" | wc -c )
#i_message="Press '${FAILURE}I${NORMAL}' to enter interactive startup"

# Set scripts to skip the file system check on reboot
#FASTBOOT=yes

# Skip reading from the console
#HEADLESS=yes

# Skip cleaning /tmp
#SKIPTMPCLEAN=yes

# For setclock
#UTC=1
#CLOCKPARAMS=

# For consolelog
#LOGLEVEL=5

# For network
#HOSTNAME=mylfs

# Delay between TERM and KILL signals at shutdown
#KILLDELAY=3

# Optional sysklogd parameters
#SYSKLOGD_PARMS="-m 0"

# Console parameters
#UNICODE=1
#KEYMAP="de-latin1"
#KEYMAP_CORRECTIONS="euro2"
#FONT="lat0-16 -m 8859-15"
#LEGACY_CHARSET=
```

7.13. Початкові файли оболонки Bash

Програма-оболонка **/bin/bash** (надалі “оболонка”) використовує колекцію початкових файлів для допомоги у створенні середовища роботи. Кожен файл має специфічне використання і може давати ефект на інтерактивні середовища по-різному. Файли у директорії `/etc` забезпечують глобальні налаштування. Якщо еквівалентний файл існує у домашній директорії, він може перезаписати глобальні налаштування.

Інтерактивна оболонка запускається після успішного входу, використовуючи **/bin/login**, для читання файлу **/etc/passwd**. Інтерактивна оболонка без входу запускається з командного рядка (наприклад, `[запрошення] $/bin/bash`). Не інтерактивні оболонки зазвичай присутні коли виконується скрипт командного рядка. Він не є інтерактивним через те, що він обробляє скрипт і не чекає на ввід користувача між командами.

Для докладнішої інформації, дивіться **info bash** у секції Bash Startup Files and Interactive Shells.

Файли `/etc/profile` і `~/.bash_profile` читаються коли оболонка запускається як інтерактивна програма.

Базовий файл `/etc/profile` нижче встановлює змінні, які необхідні для підтримки рідної мови. Належне їх встановлення приводить до:

- Вивід програм перекладається на рідну мову
- Коректна класифікація символів у букви, цифри і інші класи. Це необхідно для того, щоб bash належно приймав не ASCII символи у командному рядку на не англійських локалізаціях.
- Коректний порядок алфавітного сортування для перерахунку
- Належний розмір паперу
- Коректний формат валюти, часу і значень дати

Замініть *<ll>* нижче з дво-символьним кодом потрібної мови (наприклад, “uk”) і *<CC>* дво-буквенним кодом для відповідної країни (наприклад, “UA”). *<charmap>* повинна бути замінена на канонічну карту символів для обраної вами локалізацію. можуть бути також присутні необов'язкові модифікатори, такі як “@euro”.

Список усіх підтримуваних локалей бібліотеки Glibc може бути отриманий виконання наступної команди:

```
locale -a
```

Карти символів можуть мати псевдоніми, наприклад, на “ISO-8859-1” також посилаються як на “iso8859-1” і “iso88591”. Деякі програми не можуть обробити декілька синонімів коректно (наприклад, вимога, щоб “UTF-8”) була написана як “UTF-8”, а не “utf8”), отож це є набагато безпечніше у більшості випадках вибрати канонічне ім'я для відповідної локалі. Для визначення канонічного ім'я, виконайте наступну команду, де `<ім'я локалі>` є виводом даним командою locale -a для локалі, якій ви надаєте перевагу (“uk_UA.utf8” у нашому випадку).

```
LC_ALL=<ім'я локалі> locale charmap
```

Для локалі “uk_UA.utf8”, команда, наведена вище, виведе:

```
UTF-8
```

Це проявляється у фінальному налаштуванні “uk_UA.UTF-8”. Це є важливо, що знайдена локаль використовує евристику наведену вище перевірені до додавання їх до початкових файлів Bash:

```
LC_ALL=<ім'я_локалі> locale language
LC_ALL=<ім'я_локалі> locale charmap
LC_ALL=<ім'я_локалі> locale int_curr_symbol
LC_ALL=<ім'я_локалі> locale int_prefix
```

Команди, наведені вище, повинні виводити ім'я мови, кодування символів, яка використовується локаллю, локальну монету, і префікс для набору номера перед номером телефону. Якщо будь-яка команда з наведених вище провалиться з повідомленням подібним до наведеного нижче, це означає, що ваша локаль була не встановлена у Частині 6 чи не підтримується типовим встановленням бібліотеки Glibc.

```
locale: Cannot set LC_* to default locale: No such file or directory
```

Якщо це трапилось, ви повинні або встановити бажану локаль, використовуючи команду **localedef**, або вибрати іншу локаль. Подальші інструкції вважають, що не було таких помилок від Glibc.

Деякі пакети ззовні LFS можуть не підтримувати вибрану вами локаль. Таким прикладом буде бібліотека X (частина системи X Window System), яка виводить наступне повідомлення про помилку, якщо локаль не збігається з ніяким іменем карти символів у його внутрішніх файлах:

```
Warning: locale not supported by Xlib, locale set to C
```

У деяких випадках Xlib очікує, що карта символів буде перерахована у верхньому регістрі з канонічними тире. Для прикладу, швидше “ISO-8859-1” ніж “iso88591”. Також можливо знайти відповідні специфікації видаляючи частину карти символів специфікації локалі. Це може бути перевіреним виконуючи команду `locale charmap` у обох локалях. Для прикладу, декому необхідно змінити “de_DE.ISO-8859-15@euro” до “de_DE@euro” для того, щоб отримати підтримку цієї локалі бібліотекою Xlib.

Інші пакети можуть також працювати некоректно (але не обов'язково будуть виводити будь-які повідомлення про помилки) якщо ім'я локалі не збігається з їхніми очікуваннями. У цих випадках, розслідуючи як інші дистрибутиви підтримують вашу локаль, може даи деяку корисну інформацію.

При визначенні належного налаштування локалі, створіть файл /etc/profile:

```
cat > /etc/profile << "EOF"
# Begin /etc/profile
export LANG=<ll>_<CC>.<карта_символів><@модифікатори>
# End /etc/profile
EOF
```

Локалі “C” (типова) і “en_US” (рекомендується для користувачів США) відрізняються. “C” використовує 7-бітний набір символів US-ASCII, і вважає байти з встановленим старшим бітом помилковим символом. Через це, наприклад, команда ls заміняє їх знаком запитання у цій локалі. Також, спроба надіслати повідомлення з такими символами з Mutt чи Pine спричинить надсилання повідомлення, яке не відповідає специфікації RFC (розкладка в вихідному повідомленні буде зазначена як “uknown 8-bit”). Отож ви можете використовувати локаль “C” тільки, якщо ви впевнені, що ви ніколи не будете використовувати 8-бітні символи.

Локалі, засновані на UTF-8, добре не підтримуються багатьма програмами. Ведеться робота по документації і по можливості виправлення таких проблем, перегляньте *http://www.linuxfromscratch.org/blfs/view/svn/introduction/locale-issues.html.*

7.14 Створення файлу /etc/inputrc

Файл `/etc/inputrc` обробляє картування клавіатури для специфічних ситуацій Цей початковий файл використовується Readline — бібліотека пов'язана з введенням, яка використовується пакетом Bash і більшістю інших оболонок.

Більшість користувачів не потребують специфічного картування клавіатури, отже команди нижче створюють цільовий файл `/etc/inputrc`, який використовується більшістю користувачів, які входять в систему. Якщо ви пізніше забажаєте, що вам потрібно перезаписати типові налаштування для кожного користувача,

ви можете створити файл `.inputrc` у домашньому каталозі користувача, з модифікованим картуванням.

Для докладнішої інформації, як редагувати файл inputrc, перегляньте сторінку докементації **info bash** у секції Readline Init File. **Info readline** є також хорошим джерелом інформації.

Нижче згенеровано цільовий файл `inputrc` з коментарями для роз'яснення, що роблять деякі опції. Зверніть увагу, що коментарі не можуть бути у тих рядках що і команди. Створіть файл, використовуючи команду:

```
cat > /etc/inputrc << "EOF"
# Begin /etc/inputrc
# Modified by Chris Lynn <roryo@roryo.dynup.net>
# Allow the command prompt to wrap to the next line
set horizontal-scroll-mode Off
# Enable 8bit input
set meta-flag On
set input-meta On
# Turns off 8th bit stripping
set convert-meta Off
# Keep the 8th bit for display
set output-meta On
# немає, visible or audible
set bell-style немає
# All of the following map the escape sequence of the value
# contained in the 1st argument to the readline specific functions
"\eOd": backward-word
"\eOc": forward-word
# for linux console
"\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
# for xterm
"\eOH": beginning-of-line
"\eOF": end-of-line
# for Konsole
"\e[H": beginning-of-line
"\e[F": end-of-line
# End /etc/inputrc
EOF
```

Частина 8. Робимо систему LFS здатною до завантаження

8.1 Вступ

Настав час зробити вашу систему LFS здатною до завантаження. Ця частина обговорює створення файлу fstab, побудову ядра для нової системи LFS і встановлення завантажувача GRUB, отож система LFS може бути вибраною для завантаження під час старту комп'ютера.

8.2. Створення файлу /etc/fstab

Файл `/etc/fstab` використовується деякими програмами для визначення місця де файлові системи типово мають бути змонтовані, у якому випадку і які мають бути перевіреними (на помилки цілісності) перед монтуванням. Створіть нову таблицю файлових систем, на приклад цієї:

```
cat > /etc/fstab << "EOF"
# Початок /etc/fstab
# точка монтування файлової системи тип опції дамп fsck
# порядок
/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
# кінець /etc/fstab
EOF
```

Замініть `<xxx>`, `<yyy>` і `<fff>` значеннями відповідно до системи, для прикладу, `hda2`, `hda5` і `ext3`. Для докладнішої інформації шостого поля у цьому файлі, перегляньте сторінку **man 5 fstab**.

Файлові системи з походженням від MS-DOS чи Windows (наприклад: vfat, ntfs, smbfs, cifs, iso9660, udf) потребують опцію монтування “iocharset” для правильного інтерпретування не ASCII символів у файлових іменах. Значення цієї опції повинно бути таким як і набір символів у вашій локалі, відредагуйте його таким шляхом, щоб ядро розуміло їх. Це працює, якщо відповідні визначення наборів символів (знайдені у File systems -> Native Language Support) буди скомпільованими у ядро чи побудовані як модуль. Опція “codepage” також необхідна для файлових систем vfat і smbfs. Вона повинна бути встановленою у номер кодової сторінки, яка використовується в MS-DOC вашої країни. Наприклад, в порядку монтування флеш-пристроїв USB, користувач ru_RU.KOI8-R повинен мати наступне у опціях рядка монтування у `/etc/fstab`:

```
noauto,user,quiet,showexec,iocharset=koi8r,codepage=866
```

Відповідний фрагмент опцій для користувачів ru_RU.UTF-8 є:

```
noauto,user,quiet,showexec,iocharset=utf8,codepage=866
```

Увага

У випадках з дітерами, ядро видає наступне повідомлення:

```
FAT: utf8 is not a recommended IO charset for FAT filesystems,
filesystem will be case sensitive!
```

Ця негативна рекомендація повинна бути проігнорована, так як усі інші значення опції “iocharset”

результують у неправильному представленні фреймів у локалях UTF-8

Також можливо вказати типову кодову сторінку і набір символів для вводу/виводу для деяких файлових систем під час конфігурації ядра. Необхідні параметри названі “Default NLS Option” (`CONFIG_NLS_DEFAULT`), “Default Remote NLS Option” (`CONFIG_SMB_NLS_DEFAULT`), “Default codepage for FAT” (`CONFIG_FAT_DEFAULT_CODEPAGE`), і “Default iocharset for FAT” (`CONFIG_FAT_DEFAULT_IOCHARSET`). Не можна вказати ці налаштування для файлової системи ntfs під час компіляції ядра.

Можливо зробити файлову систему ext3 надійну при зустрічі з помилками живлення для деяких типів жорстких дисків. Щоб зробити це, додайте опцію монтування `barrier=1` до відповідного запису у файлі `/etc/fstab`. Для перевірки підтримки даної опції пристроєм, виконайте hdparam для застосованого дискового пристрою. Для приклвду, якщо:

```
hdparam -I /dev/sda | grep NCQ
```

повертає не пустий рядок, опція підтримується.

Зверніть увагу: не можна застосовувати опцію “barrier” на розділи засновані на LVM (Logical Volume Managment — логічне управління томами).

8.3. Linux-3.5.2

Пакет Linux містить ядро Лінукс.

Приблизний час побудови: 1.0 — 5.0 SBU
Необхідний дисковий простір: 540 — 800 Мбайт

8.3.1. Встановлення ядра

Побудова ядра включає в себе декілька кроків — конфігурацію, компіляцію і встановлення. Прочитайте файл README у дереві вихідного коду ядра для альтернативних методів налаштування ядра, які використовуються в даній книзі.

Підготуйте до компіляції виконуючи наступну команду:

```
make mrproper
```

Це запевняє, що дерево ядра є абсолютно чистим. Команда ядра рекомендує, щоб ця команда була виконана перед кожним компілюванням ядра. Не спирайтеся на чистоту дерева джерела коду ядра, після того як воно було розпакованим.

Налаштуйте ядро за допомогою інтерфейсу меню. Для загальної інформації про конфігурацію ядра, перегляньте сторінку за адресою *http://www.linuxfromscratch.org/hints/downloads/files/kernel-configuration.txt*. BLFS має деяку інформаціію про докладну вимоги конфігурації ядра пакету ззовні LFS за адресою http://www.linuxfromscratch. org/blfs/view/svn/longindex.html#kernel-config-index.

> Увага
>
> У зв'язку з недавніми змінами у udev, будьте впевненими вибрати
>
> Device Drivers --->
>
> Generic Driver Options --->
>
> Maintain a devtmpfs filesystem to mount at /dev

```
make LANG=<хостове_значення_LANG> LC_ALL= menuconfig
```

Значення параметрів:

LANG=<хостове_значення_LANG> LC_ALL=

Це встановлює налаштування локалі до тої, яка використовується на хості. Це необхідно для належного малювання ліній інтерфейсу menuconfig ncurses на текстових консолях UTF-8.

Впевніться у заміні <хостове_значення_LANG> значенням змінної $LANG з вашого хосту. Якщо ви її не встановили, ви можете використати заміну хостовим значенням змінних $LC_ALL чи $LC_CTYPE.

Альтернативно, **make oldconfig** може бути більш доречним у деяких ситуаціях. Дивіться файл README для більш докладної інформації.

За бажанням пропустіть конфігурацію ядра копіюванням конфігураційного файлу ядра, .config, з хостової системи (якщо такий існує) до розпакованої директорії `linux-3.5.2`. Однак, ми не рекомендуємо цю опцію. Частіше краще дослідити усі меню конфігурації і створити конфігурацію ядра з нуля. Скомпілюйте

образ ядра і його модулі:

```
make
```

При використанні модулів ядра, можливо буде потрібна конфігурація модулів у `/etc/modrobe.d`. Інформація, що відноситься до модулів і конфігурації ядра розміщена у Секції 7.4, “Обробка пристроїв і модулів на системах LFS” і у документації ядра у директорії `linux-3.5.2/Documentation`. Також, цікавим може бути `modprobe.conf(5)`.

Встановіть модулі, якщо конфігурація ядра використовує їх:

```
make modules_install
```

Після закінчення компіляції ядра, для встановлення необхідні додаткові кроки. Деякі файли необхідно скомпілювати до директорії `/boot`.

Шлях до образу ядра може відрізнятися в залежності від використовуваної платформи. Ім'я файлу, розміщеного нижче, можна змінити для задоволення ваших смаків, але основою ім'я повинно бути vmlinuz, для того щоб бути сумісним з автоматичним встановленням процесу завантаження, який описаний у наступній секції. Наступні команди припускають використання архітектури x86:

```
cp -v arch/x86/boot/bzImage /boot/vmlinuz-3.5.2-lfs-7.2
```

`System.map` є символьним файлом для ядра. Він виконує карту входу вказівників функцій у API ядра, так само як і адреси структур ядра для його виконання. Вони використовуються як джерело при розслідуванні проблем ядра. Виконайте наступні команди для встановлення файлу карти:

```
cp -v System.map /boot/System.map-3.5.2
```

Конфігураційний файл ядра `.config` згенерований попереднім з кроків **make menuconfig** містить усі конфігураційні секції для ядра, яке було тільки-що скомпільованим. Буде хорошою ідеєю зберегти цей файл для майбутніх довідок:

```
cp -v .config /boot/config-3.5.2
```

Встановіть документацію для ядра Лінукс:

```
install -d /usr/share/doc/linux-3.5.2
cp -r Documentation/* /usr/share/doc/linux-3.5.2
```

Важливо запам'ятати, що файли у директорії вихідного коду ядра не належать користувачу root. Де не були б розпакований пакет користувачом root (як ми зробили всередині root), файли мають ID користувача і групи такі самі, як іна комп'ютері, де вони були арівовані. Зазвичай це не є проблемою для інших пакетів через те, що дерево коду видаляється після встановлення. Однак, дерево коду Лінукс часто зберігається на довгий час. Через це, є шанс, що ID користувача, який архіватор прив'язав до пакету буди присвоєний декому на машині. Ця персона буде мати права запису до дерева коду ядра.

Якщо дерево коду ядра буде збережено, виконайте **chown -R 0:0** на директорії `linux-3.5.2` щоб впевнитись, що усі файли належать користувачу `root`.

Попередження

Деяка документація ядра рекомендує створення посилань з /usr/src/linux, яке буде вказувати на дерево джерела коду. Це є специфіковано для ядер попередніх версій 2.6 і не повинно бути створено на системах LFS, так як це може спричинити проблеми з пакетами, які ви бажаєте побудувати, після завершення побудови вашої системи LFS.

Попередження

Заголовкові файли у системній директорії include повинні завжди бути такими, з якими Glibc була скомпільованими, тобто, чисті заголовкові файли від архіву Лінукс. Тому, вони ніколи не повинні бути заміненими сирими заголовковими файлами ядра чи іншими файлами.

8.3.2. Налаштовування порядоку завантаження модулів Лінукс

Файл `/etc/modprobe.d/usb.conf` має бути створеним, якщо драйвери USB (ehci_hcd, ohci_hcd і uhci_hcd) були скомпільованими як модулі, вони будуть завантажені у коректному порядку; ehci_hcd потребує завантажуватися перед ohci_hcd і uhci_hcd щоб запобігти виводу попереджень при завантаженні.

Створіть новий файл `/etc/modprobe.d/usb.conf` виконуючи наступне:

```
install -v -m755 -d /etc/modprobe.d
cat > /etc/modprobe.d/usb.conf << "EOF"
# Begin /etc/modprobe.d/usb.conf
install ohci_hcd /sbin/modprobe ehci_hcd ; /sbin/modprobe -i ohci_hcd ; true
install uhci_hcd /sbin/modprobe ehci_hcd ; /sbin/modprobe -i uhci_hcd ; true
# End /etc/modprobe.d/usb.conf
EOF
```

8.3.3. Вміст Linux

Встановлені файли: config-3.5.2, vmlinux-3.5.2-lfs-7.2-3.5.2, і System.map-3.5.2

Встановлені директорії: /lib/modules, /usr/share/doc/linux-3.5.2

Короткий опис

config-3.5.2 Містить усі конфігураційні вибори для ядра

vmlinux-3.5.2-lfs-7.2 Двигун системи Лінукс. При увімкненні комп'ютера, ядро є першою частиною операційної системи, яка завантажується. Вона визначає і ініціалізує усі компоненти комп'ютерного апаратного забезпечення, після цього робить усі ці компоненти доступними як дерево файлів для програм і вмикає багатозадачну здатність процесора виконувати безліч програм ніби в один час

System.map Список адрес і символів; на які він має карту усіх вказівників і адрес усі функцій і структур даних у ядрі

8.4. Використання GRUB для встановлення процесу завантаження

8.4.1. Вступ

> Попередження
>
> Неправильне налаштування GRUB може призвести до несправності вашої системи без альтернативного завантажувального пристрою. Такого як CD-ROM. Ця секція не обов'язкова для завантаження вашої системи LFS. Ви можете просто модифікувати поточний завантажувач, наприклад Grub-Legacy, GRUB2 чи LILO.

Впевніться, що аварійний завантажувальний диск є готовим до “рятувати” комп'ютер, якщо він стане нестабільним (не здатним до завантаження). Якщо ви не не маєте завантажувальний пристрій, ви можете створити такий. У порядок роботи для нижченаведеної процедури, вам може бути необхідно стрибнути набагато вперед у BLFS і встановити **xorriso** з пакету *libisoburn*.

```
cd /tmp &&
grub-mkrescue --output=grub-img.iso &&
xorriso -as cdrecord -v dev=/dev/cdrw blank=as_needed grub-img.iso
```

8.4.2. Конвенція імен GRUB

GRUB використовує власні структури імен для пристроїв і розділів у формі (`hdn,m`), де `n` є номер приводу і m є номер розділу. Номер жорстких дисків починається з нуля, але номери розділів починаються з одиниці для нормальних розділів і з п'яти з розширених. Зверніть увагу на те, що це відрізняється від раніших версій де усі номери починалися з нуля. Для прикладу, розділи `sda1` для GRUB буде (*hd0,1*) і `sdb3` буде (*hd1,3*). В контраст Лінукс, GRUB не вважає приводи CD-ROM жорсткими приводами дисків. Для прикладу, при використанні CD на `hdb` і другого жорсткого приводу на `hdc`, цей другий жорсткий привід все-одно буде (hd1).

Ви можете визначити що GRUB думає про конфігурацію ваших дисків, виконуючи:

```
grub-mkdevicemap --device-map=device.map
cat device.map
```

8.4.3. Встановлення конфігурації

GRUB працює через те, що записує свої дані до першої фізичної доріжки на жорсткий диск. Ця зона не є частиною будь-якої файлової системи. Програми з цієї області дістаються модулів GRUB на завантажувальному розділі. Типовим розміщенням є `/boot/grub/`.

Місце знаходження завантажувального розділу є вибором користувача який виконує налаштування. Рекомендовано мати один окремий і малий розділ (приблизний розмір в 100 Мбайт) тільки для завантажувальної інформації. Для побудов таким шляхом, чи це LFS чи комерційних дистрибутив буде мати доступ до тих самих файлів завантаження і доступ може бути зроблений з будь-якої завантаженої системи. Якщо ви обрали поступити так, вам буде необхідно монтувати цей окремий розділ, перемістіть усі файли у поточній директорії `/boot` (ядро Лінукс, яке ви побудували у попередній секції) до нового розділу. Тоді вам буде необхідно від'єднати цей розділ і змотувати його знову у директорію `/boot`. Якщо ви це зробити, не забудьте оновити файл `/etc/fstab`.

Дозволено також використання поточного розділу lfs, але у такому випадку конфігурація для багатьох систем є більш складною.

Використовуючи інформацію наведену вище, визначте відповідний покажчик для розділу root (чи завантажувальний розділ, якщо такий використовується). Для наступного прикладу, вважається, що кореневий розділ є sda2.

Встановіть файли GRUB у `/boot/grub` і встановіть завантажувальну доріжку:

> Увага
>
> Наступні команди перезапишуть поточний завантажувач. Не виконуйте команду, якщо це не бажано, для прикладу, при використанні стороннього менеджера завантажень для керування MBR (Master Boot Record — головна завантажувальна доріжка).

```
grub-install /dev/sda
```

8.4.4. Створення конфігураційного файлу

Згенеруйте `/boot/grub/grub.cfg`

```
cat > /boot/grub/grub.cfg << "EOF"
# Begin /boot/grub/grub.cfg
set default=0
set timeout=5

insmod ext2
set root=(hd0,2)

menuentry "GNU/Linux, Linux 3.5.2-lfs-7.2" {
linux
/boot/vmlinuz-3.5.2-lfs-7.2 root=/dev/sda2 ro
}
EOF
```

> Увага
>
> З перспективи GRUB, файли ядра відносяться до використовуваного розділу. Якщо ви використовуєте окремий розділ для /boot, видаліть /boot з вищенаведеного рядка linux. Вам буде також необхідно змінити рядок set root для вказання розділу для завантаження.

GRUB є дуже потужною програмою і забезпечує велику кількість опцій для завантаження з широкого спектру пристроїв, операційних систем і типів розділів. Існує також багато опцій для налаштування під користувача, наприклад, графічні фони, відтворення музики, введення за допомоги мишки і так далі. Деталі цих опцій є за множиною цього вступу.

> Застереження
>
> Існує команда grub-mkconfig, яка може писати конфігураційні файли автоматично. Вона використовує набір скриптів у /etc/grub.d/ і знищить будь-які налаштування, які ви зробили. Ці скрипти розроблені головним чином для дистрибутивів, які не використовують джерела коду програм і не рекомендуються для LFS. Якщо ви встановите комерційний Лінукс дистрибутив, є хороший шанс, що ця програма буде запущеною. Не забудьте зробити резервну копію вашого файлу grub.cfg

Глава 9. Кінець

9.1 Кінець

Браво! Встановлена нова система LFS! Ми бажаємо великих успіхів з вашою новенькою налаштованою системою Лінукс.

Це буде хорошою ідеєю створити файл `/etc/lfs-release`. Маючи цей файл, буде дуже легко визначити для вас (і для нас, якщо вам необхідно спитати про допомогу), яка версія LFS встановлена на системі. Створіть цей файл виконуючи:

```
echo 7.2 > /etc/lfs-release
```

Також добре було б створити файл, щоб показати статус вашої нової системи з повагою для стандарту LSB. Для створення цього файлу, виконайте:

```
cat > /etc/lsb-release << "EOF"
DISTRIB_ID="Linux From Scratch"
DISTRIB_RELEASE="7.2"
DISTRIB_CODENAME="<your name here>"
DISTRIB_DESCRIPTION="Linux From Scratch"
EOF
```

Не забудьте внести деякі персональні дані для поля 'DIST_CODENAME' щоб зробити вашу систему унікальною.

9.2 Будьте порахованими

Тепер, коли ви закінчили книгу, чи хочете ви бути порахованим як користувач LFS? Перейдіть на сторінку *http://www.linuxfromscratch.org/cgi-bin/lfscounter.php* і зареєструйтесь як користувач LFS вводячи ваше ім'я і першу версію LFS, яку ви використали.

Давайте зараз завантажимось у нову систему LFS.

9.3 Перезавантаження системи

Тепер, після того як усі програми були встановленими, настав час перезавантажити ваш комп'ютер. Однак, ви повинні остерігатися деяких речей. Система, яку ви створили у цій книзі є мінімальною, і більш ймовірно, не буде мати функціональність яку вам буде необхідно для продовження. Встановлюючи деяких додаткових пакетів з книги BLFS з вашого поточного середовища chroot, ви можете залишити себе у набагато більш кращій позиції для продовження після того як ви перезавантажите комп'ютер у свою нову систему. Встановлюючи текстовий веб-переглядач, наприклад Lynx, ви можете легко переглядати книгу BLFS у віртуальному терміналі, поки будете будувати пакети у іншому. Пакет GPM також дозволить вам виконувати дії копіювання/встаки у вашому віртуальному терміналі. На кінець, якщо ви знаходитесь у ситуації, де статична конфігурація IP не збігається з вимогами вашої мережі, встановлення таких пакетів як Dhcpcd чи PPP також може бути корисним.

Тепер, коли ми це сказали, давайте продовжимо завантаження нашої новенької інсталяції LFS в перший раз! Для початку вийдіть з середовища chroot:

```
logout
```

Після цього від'єднайте віртуальні файлові системи:

```
umount -v $LFS/dev/pts
umount -v $LFS/dev/shm
```

```
umount -v $LFS/dev
umount -v $LFS/proc
umount -v $LFS/sys
```

Від'єднайте саму файлову систему LFS:

```
unmount -v $LFS
```

Якщо було створено нові розділи, від'єднайте інші розділи перед від'єднання головного, наприклад так:

```
umount -v $LFS/usr
umount -v $LFS/home
umount -v $LFS
```

Тепер перезавантажте систему за допомогою команди:

```
shutdouwn -r now
```

Вважаємо, що завантажувач GRUB було встановлено, як було розказано раніше, меню встановлено автоматично на пункті LFS 7.2

Коли перезавантаження завершилось, система LFS є готовою для використання і можливо додати більше програмного забезпечення для вирішення ваших завдань.

9.4. Що тепер?

Дякуємо вам за прочитання цієї книги LFS. Ми надіємось, що ви знашли цю книгу корисною і вивчили більше про процес створення системи.

Тепер коли система LFS є встановленою, ви можете задати собі питання “Що далі?”. Для відповіді на це питання, ми склали список джерел інформації для вас.

- Підтримка

  Помилки програм і нотатки безпеки рапортуються регулярно для усіх програм. Так як система LFS скомпільована з коду, на вас лежить облік таких повідомлень. Є декілька онлайн ресурсів, які збирають ці повідомлення, деякі з них наведені нижче:

  - Freshmeat.net (*http://freshmeat.net/*)

    Freshmeat може повідомляти вас (через e-mail) нові версії пакетів встановлених на вашу систему.

  - CERT (Computer Emergency Response Team)

    CERT має список розсилки, який публікує повідомлення безпеки деяких операційних систем і програм. Інформація для користувачів є доступною за адресою *http://www.us-cert.gov/cas/signup.html*.

  - Bugtraq

    Bugtraq являється комп'ютерним списком розсилки, яка повністю розкриває суть безпеки комп'ютера. Інформація для підписування на розсилання доступна за адресою *http://www.securityfocus.com/archive*.

- За межами Лінукса з початків

  Книга За межами Лінукса з початків покриває процедуру встановлення для широкого спектру програмного забезпечення за межами множини книги LFS. Проект BLFS розміщується за адресою *http://www.linuxfromscratch.org/blfs/*.

- Підказки LFS

Підказки LFS являють собою колекцією освітніх документів представленою волонтерами спілки LFS. Ці підказки доступні за адресою *http://www.linuxfromscratch.org/hints/list.html*.

- Списки розсилки

Існує декілька списків розсилки LFS на які ви можете підписатись, якщо потребуєте допомоги, чи хочете оставатись в курсі останніх розробок, хочете сприяти проекту, і більше. Дивіться Частину 1 — Списки розсилки для докладнішої інформації.

- Проект документації Лінукс

Ціль цього проекту (TLDP) у співпраці в усіх областях документації Лінукс. Проект TLDP постачає велику кількість документів HOWTO, гідів, і сторінок документації. Він знаходиться за адресою *http://www.tldp.org/*.

Частина 4. Додатки

Додаток A. Скорочення і терміни

ABI Applicaition Binary Interface — програмний бінарний інтерфейс

ALFS Автоматичний Лінукс з початків

ALSA Advanced Linux Sound Architecture — нова звукова архітектура Лінукс

API Application Programming Interface — Програмний Інтерфейс Програмування

ASCII American Standart Code for Information Interchange — Американський стандартний код для обміну інформації

BIOS Basic Input/Output Syste — Базова Система Вводу/Виводу

BLFS За межами Лінукс з Початків

BSD Berkeley Software Distribution

chroot change root зміна кореневого каталогу

CMOS Complementary Metal Oxide Semiconductor - Додатковий Металоксидний Напівпровідник

COS Class Of Service - Клас сервісу

CPU Central Processing Unit — центральна одиниця обробки

CRC Cyclic Redundancy Check — циклічна перевірка надмірності

CVS Concurent Version System — конкурента система версій

DHCP Dynamic Host Configuration Protocol — протокол динамічного конфігурування хосту

DNS Domain Name Service — сервіс доменних імен

EGA Enhanced Graphics Adapter вдосконалений графічний адаптер

ELF Executeble and Linkable Format — формат виконання і компонування

EOF End Of File — кінець файлу

EQN equation — рівняння

EVMS Enterprise Volume Management System — Комерційниа система управління томами

ext2 second extended file system — друга розширена файлова система

ext3 третя розширена файлова система

ext4	четверта розширена файлова система
FAQ	Frequently Asked Questions — питання які часто задаються
FHS	Filesystem Hierarchy Standart — стандарт файлових систем
FIFO	First-In, First Out — перший увійшов, перший вийшов
FQDN	Fully Qualified Domain Name — повне доменне ім'я
FTP	File Transfer Protocol — протокол передачі файлів
GB	Gigabytes — гігабайти
GCC	GNU Compiler Collection — колекція компіляторів GNU
GID	Group Identidier — ідентифікатор групи
GMT	Greenwich Mean Time — середній час за Грінвічем
GPG	GNU Privacy Guard — охоронець приватності GNU
HTML	Hypertext Markup Language - гіпертекстова мова розмітки
IDE	Integrated Drive Electronics - електроніка інтегрованих приводів
IEEE	Institute of Electrical and Electronic Engineers - Іститут інженерів електрики і електроніки
IO	Input/Output — ввід/вивід
IP	Internet Protocol - протокол інтернету
IPC	Inter-Process Communication міжпроцесний зв'язок
IRC	Internet Relay Chat — інтернет-чат
ISO	International Organization for Standardization - міжнародна організація по стандартизації
ISP	Internet Service Provider - провайдер сервісів інтернету
KB	Kilobytes - кілобайти, Кбайти
LED	Light Emitting Diode — світлодіоди
LFS	Linux From Scratch - Лінукс з початків
LSB	Linux Standard Base - база стандартів лінукс

MB Megabytes - мегабайти, Мбайти

MBR Master Boot Record - Головна завантажувальна доріжка

MD5 Message Digest 5 - дайджест повідомлення 5

NIC Network Interface Card - мережна інтерфейсна карта

NLS Native Language Support - підтримка рідної мови

NNTP Network News Transport Protocol - протокол передачі новин через мережу

NPTL Native POSIX Threading Library - бібліотека потоків POSIX

OSS Open Sound System - Відкрита звукова система

PCH Pre-Compiled Headers - скомпільовані заголовки

PCRE Perl Compatible Regular Expression - сумісні регулярні вирази Perl

PID Process Identifier - ідентифікатор процесу

PLFS Pure Linux From Scratch - чистий Лінукс з початків

PTY pseudo terminal — псевдотермінал

QA Quality Assurance - гарантія якості

QOS Quality Of Service - якість сервісу

RAM Random Access Memory - пам'ять випадкового доступу

RPC Remote Procedure Call - віддалений виклик процедур

RTC Real Time Clock - годинник реального часу

SBU Standard Build Unit - стандартна одиниця побудови

SCO The Santa Cruz Operation - операція Санта Круз

SGR Select Graphic Rendition - виконання вибраної графіки

SHA1 Secure-Hash Algorithm 1 - алгоритм хеш-безпеки 1

SMP Symmetric Multi-Processor - симетрична обробка на багатьох процесорах

TLDP The Linux Documentation Project - Проект документації Лінукс

TFTP Trivial File Transfer Protocol - Тривіальний протокол передачі файлів

TLS	Thread-Local Storage - Локальне збереження даних потоку
UID	User Identifier - ідентифіктор процесу
umask	user file-creation mask - маска створення файлів користувача
USB	Universal Serial Bus - універсальна послідовна шина
UTC	Coordinated Universal Time - координований універсальний час
UUID	Universally Unique Identifier - універсальний унікальний ідентифікатор
VC	Virtual Console - віртуальна консоль
VGA	Video Graphics Array - Масив відео графіки
VT	Virtual Terminal Віртуальний термінал

Додаток Б. Подяки

Ми б хотіли подякувати наступним людям і організаціям за їхні внески у проект LFS.

- Gerard Beekmans <gerard@linuxfromscratch.org> – творець LFS, лідер проекту LFS
- Matthew Burgess <matthew@linuxfromscratch.org> – Лідер проекту LFS, технічний редактор LFS
- Bruce Dubbs <bdubbs@linuxfromscratch.org> –менеджер випусків LFS, технічний редактор LFS
- Jim Gifford <jim@linuxfromscratch.org> – Один з лідерів проекту CLFS
- Bryan Kadzban <bryan@linuxfromscratch.org> – технічний редактор LFS
- Randy McMurchy <randy@linuxfromscratch.org> – лідер проекту BLFS, редактор LFS
- DJ Lucas <dj@linuxfromscratch.org> – редактор LFS і BLFS
- Ken Moffat <ken@linuxfromscratch.org> – редактор LFS і CLFS
- Ryan Oliver <ryan@linuxfromscratch.org> – один з лідерів проекту CLFS
- Безкінечна кількітсть інших людей на багатьох списках розсилки LFS і BLFS, хто допоміг зробити цю книгу можливою, даючи їхні пропозиції, тестуючи книгу і відсилаючи повідомлення про помилки, інструкції і їхній досвід у встановлені багатьох проектів.

Переклади

- Manuel Canales Esparcia <macana@macana-es.com> – Іспанський проект перекладу LFS
- Johan Lenglet <johan@linuxfromscratch.org> – Французький проект перекладу LFS
- Anderson Lizardo <lizardo@linuxfromscratch.org> – Португальський проект перекладу LFS
- Thomas Reitelbach <tr@erdfunkstelle.de> – Німецький проект перекладу LFS

Постачальники дзеркал

Північноамериканські дзеркала

- Scott Kveton <scott@osuosl.org> – дзеркало lfs.oregonstate.edu
- William Astle <lost@l-w.net> – дзеркало ca.linuxfromscratch.org
- Eujon Sellers <jpolen@rackspace.com> – дзеркало lfs.introspeed.com
- Justin Knierim <tim@idge.net> – дзеркало lfs-matrix.net

Південноамериканські дзерказа

- Manuel Canales Esparcia <manuel@linuxfromscratch.org> – дзеркало lfsmirror.lfs-es.info
- Luis Falcon <Luis Falcon> – дзеркало torredehanoi.org

Дзеркала Європи

- Guido Passet <guido@primerelay.net> – дзеркало nl.linuxfromscratch.org
- Bastiaan Jacques <baafie@planet.nl> – дзеркало lfs.pagefault.net
- Sven Cranshoff <sven.cranshoff@lineo.be> – дзеркало lfs.lineo.be
- Scarlet Belgium – дзеркало lfs.scarlet.be
- Sebastian Faulborn <info@aliensoft.org> – дзеркало lfs.aliensoft.org
- Stuart Fox <stuart@dontuse.ms> – дзеркало lfs.dontuse.ms
- Ralf Uhlemann <admin@realhost.de> – дзеркало lfs.oss-mirror.org
- Antonin Sprinzl <Antonin.Sprinzl@tuwien.ac.at> – дзеркало at.linuxfromscratch.org
- Fredrik Danerklint <fredan-lfs@fredan.org> – дзеркало se.linuxfromscratch.org
- Franck <franck@linuxpourtous.com> – дзеркало lfs.linuxpourtous.com
- Philippe Baqué <baque@cict.fr> – дзеркало lfs.cict.fr
- Vitaly Chekasin <gyouja@pilgrims.ru> – дзеркало lfs.pilgrims.ru
- Benjamin Heil <kontakt@wankoo.org> – дзеркало lfs.wankoo.org

Дзеркала Азії

- Satit Phermsawang <satit@wbac.ac.th> – дзеркало lfs.phayoune.org
- Shizunet Co.,Ltd. <info@shizu-net.jp> – дзеркало lfs.mirror.shizu-net.jp
- Init World <http://www.initworld.com/> – дзеркало lfs.initworld.com

Australian Mirrors

- Jason Andrade <jason@dstc.edu.au> – дзеркало au.linuxfromscratch.org

Колишні члени команди проекту

- Christine Barczak <theladyskye@linuxfromscratch.org> – Редактор книги LFS
- Archaic <archaic@linuxfromscratch.org> – технічний редактор LFS, лідер проекту HLFS, редактор BLFS , супроводжуючий проекту підказок і патчів
- Nathan Coulson <nathan@linuxfromscratch.org> – Супроводжуючий LFS-Bootscripts
- Timothy Bauscher
- Robert Briggs
- Ian Chilton
- Jeroen Coumans <jeroen@linuxfromscratch.org> – Розроблення сайту, супроводжуючий FAQ
- Manuel Canales Esparcia <manuel@linuxfromscratch.org> – супроводжувач LFS/BLFS/HLFS XML і XSL

- Alex Groenewoud – технічний автор LFS
- Marc Heerdink
- Jeremy Huntwork <jhuntwork@linuxfromscratch.org> – технічний автор LFS, супроводжувач LFS LiveCD
- Mark Hymers
- Seth W. Klein – Супроводжувач FAQ
- Nicholas Leippe <nicholas@linuxfromscratch.org> – супроводжувач Wiki
- Anderson Lizardo <lizardo@linuxfromscratch.org> – Супроводжувач скриптів вебсайту
- Dan Nicholson <dnicholson@linuxfromscratch.org> – Редактор LFS і BLFS
- Alexander E. Patrakov <alexander@linuxfromscratch.org> – технічний автор LFS, Інтернаціоналізація LFS
- Editor, супроводжувач LFS Live CD
- Simon Perreault
- Scot Mc Pherson <scot@linuxfromscratch.org> – Супроводжувач шлюзу NNTP LFS
- Greg Schafer <gschafer@zip.com.au> – технічний автор LFS архітектор наступного покоління 64-бітних методів побудови
- Jesse Tie-Ten-Quee – Технічний автор LFS
- James Robertson <jwrober@linuxfromscratch.org> – супроводжувач Bugzilla
- Tushar Teredesai <tushar@linuxfromscratch.org> – редактор книги BLFS, Лідер проекту Підказок і Патчів LFS
- Jeremy Utley <jeremy@linuxfromscratch.org> – технічний автор LFS, супроводжувач Bugzilla і LFS-Bootscripts
- Zack Winkles <zwinkles@gmail.com> – Технічний автор LFS

Додаток B. Залежності

Кожен пакет побудований у LFS покладається на один чи більше інших пакетів за порядком належного встановлення і побудови. Деякі пакети навіть беруть участь у кругових залежностях, що означає, що перший апкет залежить від другого, який у свою чергу залежить від першого. Через те ці залежності, порядок в якому ці пакети є побудованими у LFS є дуже важливим. Ціль цієї сторінки є в документуванні цих залежностей кожного побудованого пакету у LFS.

Для кожного пакету, який ми побудували, ми перераховуємо три, а деяких випадках чотири типи залежностей. Перший перераховує які інші пакети мають бути в наявності зоб скомпілювати і встановити пакет. Другий список залежностей, в добавок до тих у першому, мають бути присутніми для виконання тестових наборів. Третій список залежностей є пакети, які вимагають, щоб цей пакет був побудованим і встановленим перед тим як вони будуть побудовані і встановлені. Якщо не будувати пакети у певному порядку, це може виявитись у шляхах /tools/bin/[бінарник] які будуть поміщені всередині скриптів у фінальній системі. Це очевидно не бажано.

Останній список залежностей є необов'язковими пакетами, які не адресовані у LFS, але можуть бути корисними для користувача. Ці пакети можуть мати свою додаткову обов'язкову чи необов'язкову залежності. Для цих залежностей, рекомендованою практикою буде встановлення їх після завершення книги LFS і після цього повернутися до перебудови пакету LFS. У деяких випадках, повторне встановлення адресоване у BLFS.

Autoconf

Встановлення залежить на: Bash, Coreutils, Grep, M4, Make, Perl, Sed, і Texinfo

Тестовий набір залежить від: Automake, Diffutils, Findutils, GCC, і Libtool

Повинен бути встановленим перед: Automake

Необов'язкові залежності: Emacs

Automake

Встановлення залежить на: Autoconf, Bash, Coreutils, Gettext, Grep, M4, Make, Perl, Sed,і Texinfo

Тестовий набір залежить від: Binutils, Bison, Bzip2, DejaGNU, Diffutils, Expect, Findutils, Flex, GCC, Gettext, Gzip, Libtool і Tar.

Повинен бути встановленим перед: немає

Необов'язкові залежності: немає

Bash

Встановлення залежить на: Bash, Binutils, Bison, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Make, Ncurses, Patch, Readline, Sed, і Texinfo

Тестовий набір залежить від: Shadow

Повинен бути встановленим перед: немає

Необов'язкові залежності: Xorg

Binutils

Встановлення залежить на: Bash, Binutils, Coreutils, Diffutils, File, Gawk, GCC, Glibc, Grep, Make, Perl, Sed, Texinfo і Zlib

Тестовий набір залежить від: DejaGNU і Expect

Повинен бути встановленим перед: немає

Необов'язкові залежності: немає

Bison

Встановлення залежить на: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, M4, Make, і Sed

Тестовий набір залежить від: Diffutils і Findutils

Повинен бути встановленим перед: Flex, Kbd, і Tar

Необов'язкові залежності: Doxygen (тестовий набір)

Bzip2

Встановлення залежить на: Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Make, і Patch

Тестовий набір залежить від: немає

Повинен бути встановленим перед: немає

Необов'язкові залежності: немає

Coreutils

Встановлення залежить на: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, GMP, Grep, Make, Patch, Perl, Sed, і Texinfo

Тестовий набір залежить від: Diffutils, E2fsprogs, Findutils, Shadow, і Util-linux

Повинен бути встановленим перед: Bash, Diffutils, Findutils, Man-DB і Udev

Необов'язкові залежності: Perl Expect and IO:Tty modules (для тестових наборів)

DejaGNU

Встановлення залежить на: Bash, Coreutils, Diffutils, GCC, Grep, Make, і Sed

Тестовий набір залежить від: Немає доступних тестових наборівтестових наборів

Повинен бути встановленим перед: немає

Необов'язкові залежності: немає

Diffutils

Встановлення залежить на: Bash, Binutils, Coreutils, Gawk, GCC, Gettext, Glibc, Grep, Make, Sed, і Texinfo

Тестовий набір залежить від: Diffutils, Perl

Повинен бути встановленим перед: немає

Необов'язкові залежності: немає

Expect

Встановлення залежить на: Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Grep, Make, Patch, Sed, і Tcl

Тестовий набір залежить від: немає

Повинен бути встановленим перед: немає

Необов'язкові залежності: немає

E2fsprogs

Встановлення залежить на: Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Gzip, Make, Sed, Texinfo, і Util-linux

Тестовий набір залежить від: Psmisc

Повинен бути встановленим перед: немає

Необов'язкові залежності: немає

File

Встановлення залежить на: Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Make, Sed, і Zlib

Тестовий набір залежить від: немає

Повинен бути встановленим перед: немає

Необов'язкові залежності: немає

Findutils

Встановлення залежить на: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, Make, Sed, і Texinfo

Тестовий набір залежить від: DejaGNU, Diffutils, і Expect

Повинен бути встановленим перед: немає

Необов'язкові залежності: немає

Flex

Встановлення залежить на: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, M4, Make, Patch, Sed, і Texinfo

Тестовий набір залежить від: Bison і Gawk

Повинен бути встановленим перед: IPRoute2, Kbd, and Man-DB

Необов'язкові залежності: немає

Gawk

Встановлення залежить на: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, Make, Patch, Sed і Texinfo

Тестовий набір залежить від: Diffutils

Повинен бути встановленим перед: немає

Необов'язкові залежності: немає

Gcc

Встановлення залежить на: Bash, Binutils, Coreutils, Diffutils, Findutils, Gawk, GCC, Gettext, Glibc, GMP, Grep, M4, Make, MPC, MPFR, Patch, Perl, Sed, Tar, і Texinfo

Тестовий набір залежить від: DejaGNU і Expect

Повинен бути встановленим перед: немає

Необов'язкові залежності: CLooG-PPL, GNAT і PPL

GDBM

Встановлення залежить на: Bash, Binutils, Coreutils, Diffutils, GCC, Grep, Make, і Sed

Тестовий набір залежить від: немає

Повинен бути встановленим перед: немає

Необов'язкові залежності: немає

Gettext

Встановлення залежить на: Bash, Binutils, Coreutils, Gawk, GCC, Glibc, Grep, Make, Sed, і Texinfo

Тестовий набір залежить від: Diffutils, Perl, і Tcl

Повинен бути встановленим перед: Automake

Необов'язкові залежності: немає

Glibc

Встановлення залежить на: Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Gettext, Grep, Gzip, заголовкові файли Linux API , Make, Perl, Sed, і Texinfo

Тестовий набір залежить від: File

Повинен бути встановленим перед: немає

Необов'язкові залежності: немає

GMP

Встановлення залежить на: Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, M4, Make, Sed і Texinfo

Тестовий набір залежить від: немає

Повинен бути встановленим перед: MPFR, GCC

Необов'язкові залежності: немає

Grep

Встановлення залежить на: Bash, Binutils, Coreutils, Diffutils, GCC, Gettext, Glibc, Grep, Make, Patch, Sed, і Texinfo

Тестовий набір залежить від: Gawk

Повинен бути встановленим перед: Man-DB

Необов'язкові залежності: Pcre, Xorg і CUPS

Groff

Встановлення залежить на: Bash, Binutils, Bison, Coreutils, Gawk, GCC, Glibc, Grep, Make, Patch, Sed, і Texinfo

Тестовий набір залежить від: Немає доступних тестових наборів тестових наборів

Повинен бути встановленим перед: Man-DB і Perl

Необов'язкові залежності: GPL Ghostscript

GRUB

Встановлення залежить на: Bash, Binutils, Bison, Coreutils, Diffutils, GCC, Gettext, Glibc, Grep, Make, Ncurses, Sed, Texinfo, і Xz

Тестовий набір залежить від: немає

Повинен бути встановленим перед: немає

Необов'язкові залежності: немає

Gzip

Встановлення залежить на: Bash, Binutils, Coreutils, GCC, Glibc, Grep, Make, Sed і Texinfo

Тестовий набір залежить від: Diffutils. Less

Повинен бути встановленим перед: Man-DB

Необов'язкові залежності: немає

Iana-Etc

Встановлення залежить на: Coreutils, Gawk і Make

Тестовий набір залежить від: Немає доступних тестових наборів тестових наборів

Повинен бути встановленим перед: Perl

Необов'язкові залежності: немає

Inetutils

Встановлення залежить на: Bash, Binutils, Coreutils, GCC, Glibc, Grep, Make, Ncurses, Patch,

Sed, Texinfo і Zlib

Тестовий набір залежить від: Немає доступних тестових наборів тестових наборів

Повинен бути встановленим перед: Tar

Необов'язкові залежності: немає

IProute2

Встановлення залежить на: Bash, Bison, Coreutils, Flex, GCC, Glibc, Make і заголовкові файли Linux API

Тестовий набір залежить від: Немає доступних тестових наборів тестових наборів

Повинен бути встановленим перед: немає

Необов'язкові залежності: немає

Kbd

Встановлення залежить на: Bash, Binutils, Bison, Coreutils, Flex, GCC, Gettext, Glibc, Gzip, Make, Patch і Sed

Тестовий набір залежить від: Немає доступних тестових наборів тестових наборів

Повинен бути встановленим перед: немає

Необов'язкові залежності: немає

Kmod

Встановлення залежить на: Bash, Binutils, Bison, Coreutils, Flex, GCC, Gettext, Glibc, Gzip, Make, Sed, Xz-Utils, Zlib

Тестовий набір залежить від: Немає доступних тестових наборів тестових наборів

Повинен бути встановленим перед: Udev

Необов'язкові залежності: немає

Less

Встановлення залежить на: Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Grep, Make, Ncurses і Sed

Тестовий набір залежить від: Немає доступних тестових наборів тестових наборів

Повинен бути встановленим перед: Gzip

Необов'язкові залежності: Pcre

Libpipeline

Встановлення залежить на: Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Make, Sed і Texinfo

Тестовий набір залежить від: Немає доступних тестових наборів тестових наборів

Повинен бути встановленим перед: Man-DB

Необов'язкові залежності: немає

Libtool

Встановлення залежить на: Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Make, Sed і Texinfo

Тестовий набір залежить від: Findutils

Повинен бути встановленим перед: немає

Необов'язкові залежності: немає

Linux Kernel

Встановлення залежить на: Bash, Binutils, Coreutils, Diffutils, Findutils, GCC, Glibc, Grep, Gzip, Kmod, Make, Ncurses, Perl і Sed

Тестовий набір залежить від: Немає доступних тестових наборів тестових наборів

Повинен бути встановленим перед: немає

Необов'язкові залежності: немає

M4

Встановлення залежить на: Bash, Binutils, Coreutils, GCC, Glibc, Grep, Make, Sed, і Texinfo Diffutils

Тестовий набір залежить від: Diffutils

Повинен бути встановленим перед: Autoconf і Bison

Необов'язкові залежності: libsigsegv

Make

Встановлення залежить на: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, Make, Sed і Texinfo

Тестовий набір залежить від: Perl і Procps

Повинен бути встановленим перед: немає

Необов'язкові залежності: немає

Man-DB

Встановлення залежить на: Bash, Binutils, Bzip2, Coreutils, Flex, GCC, GDBM, Gettext, Glibc, Grep, Groff, Gzip, Less, Libpipeline, Make, Sed і Xz

Тестовий набір залежить від: Не виконується. Вимагає тестовий набір пакету Man-DB

Повинен бути встановленим перед: немає

Необов'язкові залежності: немає

Man-Pages

Встановлення залежить на: Bash, Coreutils і Make

Тестовий набір залежить від: Немає доступних тестових наборів тестових наборів

Повинен бути встановленим перед: немає

Необов'язкові залежності: немає

MPC

Встановлення залежить на: Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, GMP, Make, MPFR, Sed і Texinfo

Тестовий набір залежить від: немає

Повинен бути встановленим перед: GCC

Необов'язкові залежності: немає

MPFR

Встановлення залежить на: Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, GMP, Make, Sed і Texinfo

Тестовий набір залежить від: немає

Повинен бути встановленим перед: GCC

Необов'язкові залежності: немає

Ncurses

Встановлення залежить на: Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Make, Patch і Sed

Тестовий набір залежить від: Немає доступних тестових наборів тестових наборів

Повинен бути встановленим перед: Bash, GRUB, Inetutils, Less, Procps, Psmisc, Readline, Texinfo, Util-linux і Vim

Необов'язкові залежності: немає

Patch

Встановлення залежить на: Bash, Binutils, Coreutils, GCC, Glibc, Grep, Make і Sed

Тестовий набір залежить від: Немає доступних тестових наборів тестових наборів

Повинен бути встановленим перед: Bash, GRUB, Inetutils, Less, Procps, Psmisc, Readline, Texinfo, Util-linux і Vim

Необов'язкові залежності: Ed

Perl

Встановлення залежить на: Bash, Binutils, Coreutils, Gawk, GCC, GDBM, Glibc, Grep, Groff, Make, Sed і Zlib

Тестовий набір залежить від: Iana-Etc і Procps

Повинен бути встановленим перед: Autoconf

Необов'язкові залежності: немає

Pkg-config

Встановлення залежить на: Bash, Binutils, Coreutils, Gawk, GCC, Glibc, Grep, Make, Popt і Sed

Тестовий набір залежить від: немає

Повинен бути встановленим перед: Kmod

Необов'язкові залежності: немає

Popt

Встановлення залежить на: Bash, Binutils, Coreutils, Gawk, GCC, Glibc, Grep і Make

Тестовий набір залежить від: Diffutils і Sed

Повинен бути встановленим перед: Pkg-config

Необов'язкові залежності: немає

Procps

Встановлення залежить на: Bash, Binutils, Coreutils, GCC, Glibc, Make і Ncurses

Тестовий набір залежить від: Немає доступних тестових наборів тестових наборів

Повинен бути встановленим перед: немає

Необов'язкові залежності: немає

Psmisc

Встановлення залежить на: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, Make, Ncurses і Sed

Тестовий набір залежить від: Немає доступних тестових наборів тестових наборів

Повинен бути встановленим перед: немає

Необов'язкові залежності: немає

Readline

Встановлення залежить на: Bash, Binutils, Coreutils, Gawk, GCC, Glibc, Grep, Make, Ncurses, Patch, Sed і Texinfo

Тестовий набір залежить від: Немає доступних тестових наборів тестових наборів

Повинен бути встановленим перед: Bash

Необов'язкові залежності: немає

Sed

Встановлення залежить на: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, Make, Sed і Texinfo

Тестовий набір залежить від: Diffutils і Gawk

Повинен бути встановленим перед: E2fsprogs, File, Libtool і Shadow

Необов'язкові залежності: Cracklib

Shadow

Встановлення залежить на: Bash, Binutils, Coreutils, Diffutils, Findutils, Gawk, GCC, Gettext, Glibc, Grep, Make і Sed

Тестовий набір залежить від: Немає доступних тестових наборів тестових наборів

Повинен бути встановленим перед: Coreutils

Необов'язкові залежності: Acl, Attr, Cracklib, PAM

Sysklogd

Встановлення залежить на: Binutils, Coreutils, GCC, Glibc, Make і Patch

Тестовий набір залежить від: Немає доступних тестових наборів тестових наборів

Повинен бути встановленим перед: немає

Необов'язкові залежності: немає

Sysvinit

Встановлення залежить на: Binutils, Coreutils, GCC, Glibc, Make і Sed

Тестовий набір залежить від: Немає доступних тестових наборів тестових наборів

Повинен бути встановленим перед: немає

Необов'язкові залежності: немає

Tar

Встановлення залежить на: Bash, Binutils, Bison, Coreutils, GCC, Gettext, Glibc, Grep, Inetutils, Make, Sed і Texinfo

Тестовий набір залежить від: Autoconf, Diffutils, Findutils, Gawk і Gzip

Повинен бути встановленим перед: немає

Необов'язкові залежності: немає

Tcl

Встановлення залежить на: Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Grep, Make, and Sed

Тестовий набір залежить від: немає

Повинен бути встановленим перед: немає

Необов'язкові залежності: немає

Texinfo

Встановлення залежить на: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, Make, Ncurses, Patch і Sed

Тестовий набір залежить від: немає

Повинен бути встановленим перед: немає

Необов'язкові залежності: немає

Udev

Встановлення залежить на: Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Kmod, Make і Sed

Тестовий набір залежить від: Немає доступних тестових наборів тестових наборів

Повинен бути встановленим перед: немає

Необов'язкові залежності: Glib, Pci-Utils, Python, Systemd, USB-Utils

Util-linux

Встановлення залежить на: Bash, Binutils, Coreutils, Diffutils, Findutils, Gawk, GCC, Gettext, Glibc, Grep, Make, Ncurses, Sed і Zlib

Тестовий набір залежить від: Немає доступних тестових наборів тестових наборів

Повинен бути встановленим перед: немає

Необов'язкові залежності: Glib, Pci-Utils, Python, Systemd, USB-Utils

Vim

Встановлення залежить на: Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Grep, Make, Ncurses і Sed

Тестовий набір залежить від: немає

Повинен бути встановленим перед: немає

Необов'язкові залежності: Xorg, GTK+2, LessTif, Python, Tcl, Ruby, and GP

Xz

Встановлення залежить на: Bash, Binutils, Coreutils, Diffutils, GCC, Glibc і Make

Тестовий набір залежить від: немає

Повинен бути встановленим перед: GRUB, Kmod, Man-DB, Udev

Необов'язкові залежності: немає

Zlib

Встановлення залежить на: Bash, Binutils, Coreutils, GCC, Glibc, Make і Sed

Тестовий набір залежить від: немає

Повинен бути встановленим перед: File, Kmod, Perl, and Util-linux

Необов'язкові залежності: немає

Додаток Г. Завантажувальні скрипти і скрипти sysconfig веріся-20120901

Скрипти у цьому додатку до книги перераховані по директоріям де вони типово розміщуються. Порядок такий: /etc/rc.d/init.d, /etc/sysconfig, /etc/sysconfig/network-devices, і /etc/sysconfig/network-devices/services. У кожній секції, файли перераховані у порядку в якому вони типово викликаються.

Г.1. /etc/rc.d/init.d/rc

Скрипт rc є перший скрипт, який викликається програмою init і ініціалізує процес завантаження.

```
#!/bin/bash
#############################################################################
# Begin rc
#
# Description : Main Run Level Control Script
#
# Authors     : Gerard Beekmans  - gerard@linuxfromscratch.org
#             : DJ Lucas - dj@linuxfromscratch.org
# Update      : Bruce Dubbs - bdubbs@linuxfromscratch.org
#
# Version     : LFS 7.0
#
#############################################################################

. /lib/lsb/init-functions

print_error_msg()
{
   log_failure_msg
   # $i is set when called
   MSG="FAILURE:\n\nYou should not be reading this error message.\n\n"
   MSG="${MSG}It means that an unforeseen error took place in\n"
   MSG="${MSG}${i},\n"
   MSG="${MSG}which exited with a return value of ${error_value}.\n"

   MSG="${MSG}If you're able to track this error down to a bug in one of\n"
   MSG="${MSG}the files provided by the files provided by\n"
   MSG="${MSG}the ${DISTRO_MINI} book, please be so kind to inform us at\n"
   MSG="${MSG}${DISTRO_CONTACT}.\n"
   log_failure_msg "${MSG}"

   log_info_msg "Press Enter to continue..."
   wait_for_user
}

check_script_status()
{
   # $i is set when called
   if [ ! -f ${i} ]; then
      log_warning_msg "${i} is not a valid symlink."
      continue
   fi
```

```
   if [ ! -x ${i} ]; then
      log_warning_msg "${i} is not executable, skipping."
      continue
   fi
}

run()
{
   if [ -z $interactive ]; then
      ${1} ${2}
      return $?
   fi

   while true; do
      read -p "Run ${1} ${2} (Yes/no/continue)? " -n 1 runit
      echo

      case ${runit} in
         c | C)
            interactive=""
            ${i} ${2}
            ret=${?}
            break;
            ;;

         n | N)
            return 0
            ;;

         y | Y)
            ${i} ${2}
            ret=${?}
            break
            ;;
      esac
   done

   return $ret
}

# Read any local settings/overrides
[ -r /etc/sysconfig/rc.site ] && source /etc/sysconfig/rc.site

DISTRO=${DISTRO:-"Linux From Scratch"}
DISTRO_CONTACT=${DISTRO_CONTACT:-"lfs-dev@linuxfromscratch.org (Registration
required)"}
DISTRO_MINI=${DISTRO_MINI:-"LFS"}
IPROMPT=${IPROMPT:-"no"}

# These 3 signals will not cause our script to exit
trap "" INT QUIT TSTP

[ "${1}" != "" ] && runlevel=${1}
```

```
if [ "${runlevel}" == "" ]; then
   echo "Usage: ${0} <runlevel>" >&2
   exit 1
fi

previous=${PREVLEVEL}
[ "${previous}" == "" ] && previous=N

if [ ! -d /etc/rc.d/rc${runlevel}.d ]; then
   log_info_msg "/etc/rc.d/rc${runlevel}.d does not exist.\n"
   exit 1
fi

if [ "$runlevel" == "6" -o "$runlevel" == "0" ]; then IPROMPT="no"; fi

# Note: In ${LOGLEVEL:-7}, it is ':' 'dash' '7', not minus 7
if [ "$runlevel" == "S" ]; then
   [ -r /etc/sysconfig/console ] && source /etc/sysconfig/console
   dmesg -n "${LOGLEVEL:-7}"
fi

if [ "${IPROMPT}" == "yes" -a "${runlevel}" == "S" ]; then
   # The total length of the distro welcome string, without escape codes
   wlen=${wlen:-$(echo "Welcome to ${DISTRO}" | wc -c )}
   welcome_message=${welcome_message:-"Welcome to ${INFO}${DISTRO}$
{NORMAL}"}

   # The total length of the interactive string, without escape codes
   ilen=${ilen:-$(echo "Press 'I' to enter interactive startup" | wc -c )}
   i_message=${i_message:-"Press '${FAILURE}I${NORMAL}' to enter interactive
startup"}


   # dcol and icol are spaces before the message to center the message
   # on screen. itime is the amount of wait time for the user to press a key
   wcol=$(( ( ${COLUMNS} - ${wlen} ) / 2 ))
   icol=$(( ( ${COLUMNS} - ${ilen} ) / 2 ))
   itime=${itime:-"3"}

   echo -e "\n\n"
   echo -e "\\033[${wcol}G${welcome_message}"
   echo -e "\\033[${icol}G${i_message}${NORMAL}"
   echo ""
   read -t "${itime}" -n 1 interactive 2>&1 > /dev/null
fi

# Make lower case
[ "${interactive}" == "I" ] && interactive="i"
[ "${interactive}" != "i" ] && interactive=""

# Read the state file if it exists from runlevel S
[ -r /var/run/interactive ] && source /var/run/interactive

# Attempt to stop all services started by the previous runlevel,
# and killed in this runlevel
```

```
if [ "${previous}" != "N" ]; then
   for i in $(ls -v /etc/rc.d/rc${runlevel}.d/K* 2> /dev/null)
   do
      check_script_status

      suffix=${i#/etc/rc.d/rc$runlevel.d/K[0-9][0-9]}
      prev_start=/etc/rc.d/rc$previous.d/S[0-9][0-9]$suffix
      sysinit_start=/etc/rc.d/rcS.d/S[0-9][0-9]$suffix

      if [ "${runlevel}" != "0" -a "${runlevel}" != "6" ]; then
         if [ ! -f ${prev_start} -a  ! -f ${sysinit_start} ]; then
            MSG="WARNING:\n\n${i} can't be "
            MSG="${MSG}executed because it was not "
            MSG="${MSG}not started in the previous "
            MSG="${MSG}runlevel (${previous})."
            log_warning_msg "$MSG"
            continue
         fi
      fi

      run ${i} stop
      error_value=${?}

      if [ "${error_value}" != "0" ]; then print_error_msg; fi
   done
fi

if [ "${previous}" == "N" ]; then export IN_BOOT=1; fi

if [ "$runlevel" == "6" -a -n "${FASTBOOT}" ]; then
   touch /fastboot
fi


# Start all functions in this runlevel
for i in $( ls -v /etc/rc.d/rc${runlevel}.d/S* 2> /dev/null)
do
   if [ "${previous}" != "N" ]; then
      suffix=${i#/etc/rc.d/rc$runlevel.d/S[0-9][0-9]}
      stop=/etc/rc.d/rc$runlevel.d/K[0-9][0-9]$suffix
      prev_start=/etc/rc.d/rc$previous.d/S[0-9][0-9]$suffix

      [ -f ${prev_start} -a ! -f ${stop} ] && continue
   fi

   check_script_status

   case ${runlevel} in
      0|6)
         run ${i} stop
         ;;
      *)
         run ${i} start
         ;;
   esac
```

```
   error_value=${?}

   if [ "${error_value}" != "0" ]; then print_error_msg; fi
done

# Store interactive variable on switch from runlevel S and remove if not
if [ "${runlevel}" == "S" -a "${interactive}" == "i" ]; then
    echo "interactive=\"i\"" > /var/run/interactive
else
    rm -f /var/run/interactive 2> /dev/null
fi

# Copy the boot log on initial boot only
if [ "${previous}" == "N" -a  "${runlevel}" != "S" ]; then
   cat /run/var/bootlog >> /var/log/boot.log

   # Mark the end of boot
   echo "--------" >> /var/log/boot.log

   # Remove the temporary file
   rm -f /run/var/bootlog 2> /dev/null
fi

# End rc
```

Г.2. /lib/lsb/init-functions

```
#!/bin/sh
########################################################################
#
# Begin /lib/lsb/init-funtions
#
# Description : Run Level Control Functions
#
# Authors     : Gerard Beekmans - gerard@linuxfromscratch.org
#             : DJ Lucas - dj@linuxfromscratch.org
# Update      : Bruce Dubbs - bdubbs@linuxfromscratch.org
#
# Version     : LFS 7.0
#
# Notes       : With code based on Matthias Benkmann's simpleinit-msb
#               http://winterdrache.de/linux/newboot/index.html
#
#               The file should be located in /lib/lsb
#
########################################################################

## Environmental setup
# Setup default values for environment
umask 022
export PATH="/bin:/usr/bin:/sbin:/usr/sbin"

## Screen Dimensions
# Find current screen size
```

```
if [ -z "${COLUMNS}" ]; then
   COLUMNS=$(stty size)
   COLUMNS=${COLUMNS##* }
fi

# When using remote connections, such as a serial port, stty size returns 0
if [ "${COLUMNS}" = "0" ]; then
   COLUMNS=80
fi

## Measurements for positioning result messages
COL=$((${COLUMNS} - 8))
WCOL=$((${COL} - 2))

## Set Cursor Position Commands, used via echo
SET_COL="\\033[${COL}G"      # at the $COL char
SET_WCOL="\\033[${WCOL}G"    # at the $WCOL char
CURS_UP="\\033[1A\\033[0G"   # Up one line, at the 0'th char
CURS_ZERO="\\033[0G"

## Set color commands, used via echo
# Please consult `man console_codes for more information
# under the "ECMA-48 Set Graphics Rendition" section
#
# Warning: when switching from a 8bit to a 9bit font,
# the linux console will reinterpret the bold (1;) to
# the top 256 glyphs of the 9bit font.  This does
# not affect framebuffer consoles

NORMAL="\\033[0;39m"         # Standard console grey
SUCCESS="\\033[1;32m"        # Success is green
WARNING="\\033[1;33m"        # Warnings are yellow
FAILURE="\\033[1;31m"        # Failures are red
INFO="\\033[1;36m"           # Information is light cyan
BRACKET="\\033[1;34m"        # Brackets are blue

# Use a colored prefix
BMPREFIX="     "
SUCCESS_PREFIX="${SUCCESS}  *  ${NORMAL}"
FAILURE_PREFIX="${FAILURE}*****${NORMAL}"
WARNING_PREFIX="${WARNING} *** ${NORMAL}"

SUCCESS_SUFFIX="${BRACKET}[${SUCCESS}  OK  ${BRACKET}]${NORMAL}"
FAILURE_SUFFIX="${BRACKET}[${FAILURE} FAIL ${BRACKET}]${NORMAL}"
WARNING_SUFFIX="${BRACKET}[${WARNING} WARN ${BRACKET}]${NORMAL}"

BOOTLOG=/run/var/bootlog
KILLDELAY=3

# Set any user specified environment variables e.g. HEADLESS
[ -r /etc/sysconfig/rc.site ]  && . /etc/sysconfig/rc.site

################################################################################
####
# start_daemon()
```

```
#
# Usage: start_daemon [-f] [-n nicelevel] [-p pidfile] pathname [args...]
#
#
#
# Purpose: This runs the specified program as a daemon
#
#
#
# Inputs: -f: (force) run the program even if it is already running.
#
#         -n nicelevel: specify a nice level. See 'man nice(1)'.
#
#         -p pidfile: use the specified file to determine PIDs.
#
#         pathname: the complete path to the specified program
#
#         args: additional arguments passed to the program (pathname)
#
#
#
# Return values (as defined by LSB exit codes):
#
#       0 - program is running or service is OK
#
#       1 - generic or unspecified error
#
#       2 - invalid or excessive argument(s)
#
#       5 - program is not installed
#
##############################################################################
####
start_daemon()
{
    local force=""
    local nice="0"
    local pidfile=""
    local pidlist=""
    local retval=""

    # Process arguments
    while true
    do
        case "${1}" in

            -f)
                force="1"
                shift 1
                ;;

            -n)
                nice="${2}"
                shift 2
                ;;
```

```
            -p)
                pidfile="${2}"
                shift 2
                ;;

            -*)
                return 2
                ;;

            *)
                program="${1}"
                break
                ;;
        esac
    done

    # Check for a valid program
    if [ ! -e "${program}" ]; then return 5; fi

    # Execute
    if [ -z "${force}" ]; then
        if [ -z "${pidfile}" ]; then
            # Determine the pid by discovery
            pidlist=`pidofproc "${1}"`
            retval="${?}"
        else
            # The PID file contains the needed PIDs
            # Note that by LSB requirement, the path must be given to
pidofproc,
            # however, it is not used by the current implementation or
standard.
            pidlist=`pidofproc -p "${pidfile}" "${1}"`
            retval="${?}"
        fi

        # Return a value ONLY
        # It is the init script's (or distribution's functions)
responsibilty
        # to log messages!
        case "${retval}" in

            0)
                # Program is already running correctly, this is a
                # successful start.
                return 0
                ;;

            1)
                # Program is not running, but an invalid pid file exists
                # remove the pid file and continue
                rm -f "${pidfile}"
                ;;

            3)
```

```
                # Program is not running and no pidfile exists
                # do nothing here, let start_deamon continue.
                ;;

            *)
                # Others as returned by status values shall not be
interpreted
                # and returned as an unspecified error.
                return 1
                ;;
        esac
    fi

    # Do the start!
    nice -n "${nice}" "${@}"
}

########################################################################
####
# killproc()
#
# Usage: killproc [-p pidfile] pathname [signal]
#
#
#
# Purpose: Send control signals to running processes
#
#
#
# Inputs: -p pidfile, uses the specified pidfile
#
#         pathname, pathname to the specified program
#
#         signal, send this signal to pathname
#
#
#
# Return values (as defined by LSB exit codes):
#
#       0 - program (pathname) has stopped/is already stopped or a
#
#           running program has been sent specified signal and stopped
#
#           successfully
#
#       1 - generic or unspecified error
#
#       2 - invalid or excessive argument(s)
#
#       5 - program is not installed
#
#       7 - program is not running and a signal was supplied
#
########################################################################
####
```

```
killproc()
{
    local pidfile
    local program
    local prefix
    local progname
    local signal="-TERM"
    local fallback="-KILL"
    local nosig
    local pidlist
    local retval
    local pid
    local delay="30"
    local piddead
    local dtime

    # Process arguments
    while true; do
        case "${1}" in
            -p)
                pidfile="${2}"
                shift 2
                ;;

             *)
                 program="${1}"
                 if [ -n "${2}" ]; then
                     signal="${2}"
                     fallback=""
                 else
                     nosig=1
                 fi

                 # Error on additional arguments
                 if [ -n "${3}" ]; then
                     return 2
                 else
                     break
                 fi
                 ;;
        esac
    done

    # Check for a valid program
    if [ ! -e "${program}" ]; then return 5; fi

    # Check for a valid signal
    check_signal "${signal}"
    if [ "${?}" -ne "0" ]; then return 2; fi

    # Get a list of pids
    if [ -z "${pidfile}" ]; then
        # determine the pid by discovery
        pidlist=`pidofproc "${1}"`
        retval="${?}"
```

```
    else
        # The PID file contains the needed PIDs
        # Note that by LSB requirement, the path must be given to pidofproc,
        # however, it is not used by the current implementation or standard.
        pidlist=`pidofproc -p "${pidfile}" "${1}"`
        retval="${?}"
    fi

    # Return a value ONLY
    # It is the init script's (or distribution's functions) responsibilty
    # to log messages!
    case "${retval}" in

        0)
            # Program is running correctly
            # Do nothing here, let killproc continue.
            ;;

        1)
            # Program is not running, but an invalid pid file exists
            # Remove the pid file.
            rm -f "${pidfile}"

            # This is only a success if no signal was passed.
            if [ -n "${nosig}" ]; then
                return 0
            else
                return 7
            fi
            ;;

        3)
            # Program is not running and no pidfile exists
            # This is only a success if no signal was passed.
            if [ -n "${nosig}" ]; then
                return 0
            else
                return 7
            fi
            ;;

        *)
            # Others as returned by status values shall not be interpreted
            # and returned as an unspecified error.
            return 1
            ;;
    esac

    # Perform different actions for exit signals and control signals
    check_sig_type "${signal}"

    if [ "${?}" -eq "0" ]; then # Signal is used to terminate the program

        # Account for empty pidlist (pid file still exists and no
        # signal was given)
```

```
        if [ "${pidlist}" != "" ]; then

            # Kill the list of pids
            for pid in ${pidlist}; do

                kill -0 "${pid}" 2> /dev/null

                if [ "${?}" -ne "0" ]; then
                    # Process is dead, continue to next and assume all is
well
                    continue
                else
                    kill "${signal}" "${pid}" 2> /dev/null

                    # Wait up to ${delay}/10 seconds to for "${pid}" to
                    # terminate in 10ths of a second

                    while [ "${delay}" -ne "0" ]; do
                        kill -0 "${pid}" 2> /dev/null || piddead="1"
                        if [ "${piddead}" = "1" ]; then break; fi
                        sleep 0.1
                        delay="$(( ${delay} - 1 ))"
                    done

                    # If a fallback is set, and program is still running,
then
                    # use the fallback
                    if [ -n "${fallback}" -a "${piddead}" != "1" ]; then
                        kill "${fallback}" "${pid}" 2> /dev/null
                        sleep 1
                        # Check again, and fail if still running
                        kill -0 "${pid}" 2> /dev/null && return 1
                    else
                        # just check one last time and if still alive, fail
                        sleep 1
                        kill -0 "${pid}" 2> /dev/null && return 1
                    fi
                fi
            done
        fi

        # Check for and remove stale PID files.
        if [ -z "${pidfile}" ]; then
            # Find the basename of $program
            prefix=`echo "${program}" | sed 's/[^/]*$//'`
            progname=`echo "${program}" | sed "s@${prefix}@@"`

            if [ -e "/var/run/${progname}.pid" ]; then
                rm -f "/var/run/${progname}.pid" 2> /dev/null
            fi
        else
            if [ -e "${pidfile}" ]; then rm -f "${pidfile}" 2> /dev/null; fi
        fi

    # For signals that do not expect a program to exit, simply
```

```
    # let kill do it's job, and evaluate kills return for value

    else # check_sig_type - signal is not used to terminate program
        for pid in ${pidlist}; do
            kill "${signal}" "${pid}"
            if [ "${?}" -ne "0" ]; then return 1; fi
        done
    fi
}

################################################################################
####
# pidofproc()
#
# Usage: pidofproc [-p pidfile] pathname
#
#
#
# Purpose: This function returns one or more pid(s) for a particular daemon
#
#
#
# Inputs: -p pidfile, use the specified pidfile instead of pidof
#
#         pathname, path to the specified program
#
#
#
# Return values (as defined by LSB status codes):
#
#       0 - Success (PIDs to stdout)
#
#       1 - Program is dead, PID file still exists (remaining PIDs output)
#
#       3 - Program is not running (no output)
#
################################################################################
####
pidofproc()
{
    local pidfile
    local program
    local prefix
    local progname
    local pidlist
    local lpids
    local exitstatus="0"

    # Process arguments
    while true; do
        case "${1}" in

            -p)
                pidfile="${2}"
                shift 2
```

```
            ;;

        *)
            program="${1}"
            if [ -n "${2}" ]; then
                # Too many arguments
                # Since this is status, return unknown
                return 4
            else
                break
            fi
            ;;
    esac
done

# If a PID file is not specified, try and find one.
if [ -z "${pidfile}" ]; then
    # Get the program's basename
    prefix=`echo "${program}" | sed 's/[^/]*$//'`

    if [ -z "${prefix}" ]; then
       progname="${program}"
    else
       progname=`echo "${program}" | sed "s@${prefix}@@"`
    fi

    # If a PID file exists with that name, assume that is it.
    if [ -e "/var/run/${progname}.pid" ]; then
        pidfile="/var/run/${progname}.pid"
    fi
fi

# If a PID file is set and exists, use it.
if [ -n "${pidfile}" -a -e "${pidfile}" ]; then

    # Use the value in the first line of the pidfile
    pidlist=`/bin/head -n1 "${pidfile}"`
    # This can optionally be written as 'sed 1q' to repalce 'head -n1'
    # should LFS move /bin/head to /usr/bin/head
else
    # Use pidof
    pidlist=`pidof "${program}"`
fi

# Figure out if all listed PIDs are running.
for pid in ${pidlist}; do
    kill -0 ${pid} 2> /dev/null

    if [ "${?}" -eq "0" ]; then
        lpids="${lpids}${pid} "
    else
        exitstatus="1"
    fi
done
```

```
    if [ -z "${lpids}" -a ! -f "${pidfile}" ]; then
        return 3
    else
        echo "${lpids}"
        return "${exitstatus}"
    fi
}

################################################################################
####
# statusproc()
#
# Usage: statusproc [-p pidfile] pathname
#
#
#
# Purpose: This function prints the status of a particular daemon to stdout
#
#
#
# Inputs: -p pidfile, use the specified pidfile instead of pidof
#
#         pathname, path to the specified program
#
#
#
# Return values:
#
#       0 - Status printed
#
#       1 - Input error. The daemon to check was not specified.
#
################################################################################
####
statusproc()
{
   local pidfile
   local pidlist

   if [ "${#}" = "0" ]; then
      echo "Usage: statusproc [-p pidfle] {program}"
      exit 1
   fi

   # Process arguments
   while true; do
       case "${1}" in

           -p)
               pidfile="${2}"
               shift 2
               ;;

           *)
               if [ -n "${2}" ]; then
```

```
                    echo "Too many arguments"
                    return 1
                else
                    break
                fi
                ;;
        esac
    done

    if [ -n "${pidfile}" ]; then
       pidlist=`pidofproc -p "${pidfile}" $@`
    else
       pidlist=`pidofproc $@`
    fi

    # Trim trailing blanks
    pidlist=`echo "${pidlist}" | sed -r 's/ +$//'`

    base="${1##*/}"

    if [ -n "${pidlist}" ]; then
       echo -e "${INFO}${base} is running with Process" \
          "ID(s) ${pidlist}.${NORMAL}"
    else
       if [ -n "${base}" -a -e "/var/run/${base}.pid" ]; then
          echo -e "${WARNING}${1} is not running but" \
             "/var/run/${base}.pid exists.${NORMAL}"
       else
          if [ -n "${pidfile}" -a -e "${pidfile}" ]; then
             echo -e "${WARNING}${1} is not running" \
                "but ${pidfile} exists.${NORMAL}"
          else
             echo -e "${INFO}${1} is not running.${NORMAL}"
          fi
       fi
    fi
}

########################################################################
####
# 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: $@ - Message
#
#
#
# Return values: Not used
#
########################################################################
####
log_success_msg()
{
    echo -n -e "${BMPREFIX}${@}"
    echo -e "${CURS_ZERO}${SUCCESS_PREFIX}${SET_COL}${SUCCESS_SUFFIX}"

    # Strip non-printable characters from log file
    local logmessage=`echo "${@}" | sed 's/\\\033[^a-zA-Z]*.//g'`

    timespec
    echo -e "${STAMP} ${logmessage} OK" >> ${BOOTLOG}

    return 0
}

log_success_msg2()
{
    echo -n -e "${BMPREFIX}${@}"
    echo -e "${CURS_ZERO}${SUCCESS_PREFIX}${SET_COL}${SUCCESS_SUFFIX}"

    echo " OK" >> ${BOOTLOG}

    return 0
}

########################################################################
####
# log_failure_msg()
#
```

```
# Usage: log_failure_msg ["message"]
#
#
#
# Purpose: Print a failure status message to the screen and
#
#          a boot log file.
#
#
#
# Inputs: $@ - Message
#
#
#
# Return values: Not used
#
########################################################################
####
log_failure_msg()
{
    echo -n -e "${BMPREFIX}${@}"
    echo -e "${CURS_ZERO}${FAILURE_PREFIX}${SET_COL}${FAILURE_SUFFIX}"

    # Strip non-printable characters from log file

    timespec
    local logmessage=`echo "${@}" | sed 's/\\\033[^a-zA-Z]*.//g'`
    echo -e "${STAMP} ${logmessage} FAIL" >> ${BOOTLOG}

    return 0
}

log_failure_msg2()
{
    echo -n -e "${BMPREFIX}${@}"
    echo -e "${CURS_ZERO}${FAILURE_PREFIX}${SET_COL}${FAILURE_SUFFIX}"

    echo "FAIL" >> ${BOOTLOG}

    return 0
}

########################################################################
####
# log_warning_msg()
#
# Usage: log_warning_msg ["message"]
#
#
#
# Purpose: Print a warning status message to the screen and
#
#          a boot log file.
#
#
```

```
#
# Return values: Not used
#
########################################################################
####
log_warning_msg()
{
    echo -n -e "${BMPREFIX}${@}"
    echo -e "${CURS_ZERO}${WARNING_PREFIX}${SET_COL}${WARNING_SUFFIX}"

    # Strip non-printable characters from log file
    local logmessage=`echo "${@}" | sed 's/\\\033[^a-zA-Z]*.//g'`
    timespec
    echo -e "${STAMP} ${logmessage} WARN" >> ${BOOTLOG}

    return 0
}

########################################################################
####
# log_info_msg()
#
# Usage: log_info_msg message
#
#
#
# Purpose: Print an information message to the screen and
#
#          a boot log file.  Does not print a trailing newline character.
#
#
#
# Return values: Not used
#
########################################################################
####
log_info_msg()
{
    echo -n -e "${BMPREFIX}${@}"

    # Strip non-printable characters from log file
    local logmessage=`echo "${@}" | sed 's/\\\033[^a-zA-Z]*.//g'`
    timespec
    echo -n -e "${STAMP} ${logmessage}" >> ${BOOTLOG}

    return 0
}

log_info_msg2()
{
    echo -n -e "${@}"

    # Strip non-printable characters from log file
    local logmessage=`echo "${@}" | sed 's/\\\033[^a-zA-Z]*.//g'`
    echo -n -e "${logmessage}" >> ${BOOTLOG}
```

```
    return 0
}

########################################################################
####
# evaluate_retval()
#
# Usage: Evaluate a return value and print success or failyure as
appropriate  #
#
#
# Purpose: Convenience function to terminate an info message
#
#
#
# Return values: Not used
#
########################################################################
####
evaluate_retval()
{
   local error_value="${?}"

   if [ ${error_value} = 0 ]; then
      log_success_msg2
   else
      log_failure_msg2
   fi
}

########################################################################
####
# check_signal()
#
# Usage: check_signal [ -{signal} | {signal} ]
#
#
#
# Purpose: Check for a valid signal.  This is not defined by any LSB draft,
#
#          however, it is required to check the signals to determine if the
#
#          signals chosen are invalid arguments to the other functions.
#
#
#
# Inputs: Accepts a single string value in the form or -{signal} or {signal}
#
#
#
# Return values:
#
#       0 - Success (signal is valid
#
```

```
#       1 - Signal is not valid
#
#########################################################################
####
check_signal()
{
    local valsig

    # Add error handling for invalid signals
    valsig="-ALRM -HUP -INT -KILL -PIPE -POLL -PROF -TERM -USR1 -USR2"
    valsig="${valsig} -VTALRM -STKFLT -PWR -WINCH -CHLD -URG -TSTP -TTIN"
    valsig="${valsig} -TTOU -STOP -CONT -ABRT -FPE -ILL -QUIT -SEGV -TRAP"
    valsig="${valsig} -SYS -EMT -BUS -XCPU -XFSZ -0 -1 -2 -3 -4 -5 -6 -8 -9"
    valsig="${valsig} -11 -13 -14 -15"

    echo "${valsig}" | grep -- " ${1} " > /dev/null

    if [ "${?}" -eq "0" ]; then
        return 0
    else
        return 1
    fi
}

#########################################################################
####
# check_sig_type()
#
# Usage: check_signal [ -{signal} | {signal} ]
#
#
#
# Purpose: Check if signal is a program termination signal or a control
signal #
#          This is not defined by any LSB draft, however, it is required to
#
#          check the signals to determine if they are intended to end a
#
#          program or simply to control it.
#
#
#
# Inputs: Accepts a single string value in the form or -{signal} or {signal}
#
#
#
# Return values:
#
#       0 - Signal is used for program termination
#
#       1 - Signal is used for program control
#
#########################################################################
####
check_sig_type()
```

```
{
    local valsig

    # The list of termination signals (limited to generally used items)
    valsig="-ALRM -INT -KILL -TERM -PWR -STOP -ABRT -QUIT -2 -3 -6 -9 -14
-15"

    echo "${valsig}" | grep -- " ${1} " > /dev/null

    if [ "${?}" -eq "0" ]; then
        return 0
    else
        return 1
    fi
}

##############################################################################
####
# wait_for_user()
#
#
#
# Purpose: Wait for the user to respond if not a headless system
#
#
#
##############################################################################
####
wait_for_user()
{
   # Wait for the user by default
   [ "${HEADLESS=0}" = "0" ] && read ENTER
   return 0
}

##############################################################################
####
# is_true()
#
#
#
# Purpose: Utility to test if a variable is true | yes | 1
#
#
#
##############################################################################
####
is_true()
{
   [ "$1" = "1" ] || [ "$1" = "yes" ] || [ "$1" = "true" ] ||  [ "$1" =
"y" ] ||
   [ "$1" = "t" ]
}

# End /lib/lsb/init-functions
```

Г.3. /etc/rc.d/init.d/functions

```
#!/bin/sh
########################################################################
# Begin boot functions
#
# Description : Run Level Control Functions
#
# Authors     : Gerard Beekmans - gerard@linuxfromscratch.org
# Update      : Bruce Dubbs - bdubbs@linuxfromscratch.org
#
# Version     : LFS 7.0
#
# Notes       : With code based on Matthias Benkmann's simpleinit-msb
#               http://winterdrache.de/linux/newboot/index.html
#
#               This file is only present for backward BLFS compatibility
#
########################################################################

## Environmental setup
# Setup default values for environment
umask 022
export PATH="/bin:/usr/bin:/sbin:/usr/sbin"

# Signal sent to running processes to refresh their configuration
RELOADSIG="HUP"

# Number of seconds between STOPSIG and FALLBACK when stopping processes
KILLDELAY="3"

## Screen Dimensions
# Find current screen size
if [ -z "${COLUMNS}" ]; then
   COLUMNS=$(stty size)
   COLUMNS=${COLUMNS##* }
fi

# When using remote connections, such as a serial port, stty size returns 0
if [ "${COLUMNS}" = "0" ]; then
   COLUMNS=80
fi

## Measurements for positioning result messages
COL=$((${COLUMNS} - 8))
WCOL=$((${COL} - 2))

## Provide an echo that supports -e and -n
# If formatting is needed, $ECHO should be used
case "`echo -e -n test`" in
   -[en]*)
      ECHO=/bin/echo
      ;;
   *)
      ECHO=echo
```

```
      ;;
esac

## Set Cursor Position Commands, used via $ECHO
SET_COL="\\033[${COL}G"      # at the $COL char
SET_WCOL="\\033[${WCOL}G"    # at the $WCOL char
CURS_UP="\\033[1A\\033[0G"   # Up one line, at the 0'th char

## Set color commands, used via $ECHO
# Please consult `man console_codes for more information
# under the "ECMA-48 Set Graphics Rendition" section
#
# Warning: when switching from a 8bit to a 9bit font,
# the linux console will reinterpret the bold (1;) to
# the top 256 glyphs of the 9bit font.  This does
# not affect framebuffer consoles
NORMAL="\\033[0;39m"         # Standard console grey
SUCCESS="\\033[1;32m"        # Success is green
WARNING="\\033[1;33m"        # Warnings are yellow
FAILURE="\\033[1;31m"        # Failures are red
INFO="\\033[1;36m"           # Information is light cyan
BRACKET="\\033[1;34m"        # Brackets are blue

STRING_LENGTH="0"   # the length of the current message

#************************************************************************
****
# Function - boot_mesg()
#
# Purpose:      Sending information from bootup scripts to the console
#
# Inputs:       $1 is the message
#               $2 is the colorcode for the console
#
# Outputs:      Standard Output
#
# Dependencies: - sed for parsing strings.
#          - grep for counting string length.
#
# Todo:
#************************************************************************
****
boot_mesg()
{
   local ECHOPARM=""

   while true
   do
      case "${1}" in
         -n)
            ECHOPARM=" -n "
            shift 1
            ;;
         -*)
            echo "Unknown Option: ${1}"
```

```
            return 1
            ;;
         *)
            break
            ;;
      esac
   done

   ## Figure out the length of what is to be printed to be used
   ## for warning messages.
   STRING_LENGTH=$((${#1} + 1))

   # Print the message to the screen
   ${ECHO} ${ECHOPARM} -e "${2}${1}"

   # Log the message
   [ -d /run/var ] || return
   ${ECHO} ${ECHOPARM} -e "${2}${1}" >> /run/var/bootlog
}

boot_mesg_flush()
{
   # Reset STRING_LENGTH for next message
   STRING_LENGTH="0"
}

echo_ok()
{
   ${ECHO} -n -e "${CURS_UP}${SET_COL}${BRACKET}[${SUCCESS}  OK  $
{BRACKET}]"
   ${ECHO} -e "${NORMAL}"
   boot_mesg_flush

   [ -d /run/var ] || return
   ${ECHO} -e "[ OK ]" >> /run/var/bootlog
}

echo_failure()
{
   ${ECHO} -n -e "${CURS_UP}${SET_COL}${BRACKET}[${FAILURE} FAIL $
{BRACKET}]"
   ${ECHO} -e "${NORMAL}"
    boot_mesg_flush

   [ -d /run/var ] || return
   ${ECHO} -e "[ FAIL]"  >> /run/var/bootlog
}

echo_warning()
{
   ${ECHO} -n -e "${CURS_UP}${SET_COL}${BRACKET}[${WARNING} WARN $
{BRACKET}]"
   ${ECHO} -e "${NORMAL}"
   boot_mesg_flush
```

```
   [ -d /run/var ] || return
   ${ECHO} -e "[ WARN ]"  >> /run/var/bootlog
}

echo_skipped()
{
   ${ECHO} -n -e "${CURS_UP}${SET_COL}${BRACKET}[${WARNING} SKIP $
{BRACKET}]"
   ${ECHO} -e "${NORMAL}"
   boot_mesg_flush

   [ -d /run/var ] || return
   ${ECHO} -e "  [ SKIP ]" >> /run/var/bootlog
}

wait_for_user()
{
   # Wait for the user by default
   [ "${HEADLESS=0}" = "0" ] && read ENTER
}

evaluate_retval()
{
   error_value="${?}"

   if [ ${error_value} = 0 ]; then
      echo_ok
   else
      echo_failure
   fi

   # This prevents the 'An Unexpected Error Has Occurred' from trivial
   # errors.
   return 0
}

print_status()
{
   if [ "${#}" = "0" ]; then
      echo "Usage: ${0} {success|warning|failure}"
      return 1
   fi

   case "${1}" in

      success)
         echo_ok
         ;;

      warning)
         # Leave this extra case in because old scripts
         # may call it this way.
         case "${2}" in
            running)
               ${ECHO} -e -n "${CURS_UP}"
```

```
               ${ECHO} -e -n "\\033[${STRING_LENGTH}G   "
               boot_mesg "Already running." ${WARNING}
               echo_warning
               ;;
            not_running)
               ${ECHO} -e -n "${CURS_UP}"
               ${ECHO} -e -n "\\033[${STRING_LENGTH}G   "
               boot_mesg "Not running." ${WARNING}
               echo_warning
               ;;
            not_available)
               ${ECHO} -e -n "${CURS_UP}"
               ${ECHO} -e -n "\\033[${STRING_LENGTH}G   "
               boot_mesg "Not available." ${WARNING}
               echo_warning
               ;;
            *)
               # This is how it is supposed to
               # be called
               echo_warning
               ;;
         esac
      ;;

      failure)
         echo_failure
      ;;

   esac

}

reloadproc()
{
   local pidfile=""
   local failure=0

   while true
   do
      case "${1}" in
         -p)
            pidfile="${2}"
            shift 2
            ;;
         -*)
            log_failure_msg "Unknown Option: ${1}"
            return 2
            ;;
         *)
            break
            ;;
      esac
   done

   if [ "${#}" -lt "1" ]; then
```

```
      log_failure_msg "Usage: reloadproc [-p pidfile] pathname"
      return 2
   fi

   # This will ensure compatibility with previous LFS Bootscripts
   if [ -n "${PIDFILE}" ]; then
      pidfile="${PIDFILE}"
   fi

   # Is the process running?
   if [ -z "${pidfile}" ]; then
      pidofproc -s "${1}"
   else
      pidofproc -s -p "${pidfile}" "${1}"
   fi

   # Warn about stale pid file
   if [ "$?" = 1 ]; then
      boot_mesg -n "Removing stale pid file: ${pidfile}. " ${WARNING}
      rm -f "${pidfile}"
   fi

   if [ -n "${pidlist}" ]; then
      for pid in ${pidlist}
      do
         kill -"${RELOADSIG}" "${pid}" || failure="1"
      done

      (exit ${failure})
      evaluate_retval

   else
      boot_mesg "Process ${1} not running." ${WARNING}
      echo_warning
   fi
}

statusproc()
{
   local pidfile=""
   local base=""
   local ret=""

   while true
   do
      case "${1}" in
         -p)
            pidfile="${2}"
            shift 2
            ;;
         -*)
            log_failure_msg "Unknown Option: ${1}"
            return 2
            ;;
         *)
```

```
            break
            ;;
      esac
   done

   if [ "${#}" != "1" ]; then
      shift 1
      log_failure_msg "Usage: statusproc [-p pidfile] pathname"
      return 2
   fi

   # Get the process basename
   base="${1##*/}"

   # This will ensure compatibility with previous LFS Bootscripts
   if [ -n "${PIDFILE}" ]; then
      pidfile="${PIDFILE}"
   fi

   # Is the process running?
   if [ -z "${pidfile}" ]; then
      pidofproc -s "${1}"
   else
      pidofproc -s -p "${pidfile}" "${1}"
   fi

   # Store the return status
   ret=$?

   if [ -n "${pidlist}" ]; then
      ${ECHO} -e "${INFO}${base} is running with Process"\
         "ID(s) ${pidlist}.${NORMAL}"
   else
      if [ -n "${base}" -a -e "/var/run/${base}.pid" ]; then
         ${ECHO} -e "${WARNING}${1} is not running but"\
            "/var/run/${base}.pid exists.${NORMAL}"
      else
         if [ -n "${pidfile}" -a -e "${pidfile}" ]; then
            ${ECHO} -e "${WARNING}${1} is not running"\
               "but ${pidfile} exists.${NORMAL}"
         else
            ${ECHO} -e "${INFO}${1} is not running.${NORMAL}"
         fi
      fi
   fi

   # Return the status from pidofproc
   return $ret
}

# The below functions are documented in the LSB-generic 2.1.0

#**********************************************************************
****
# Function - pidofproc [-s] [-p pidfile] pathname
```

```
#
# Purpose: This function returns one or more pid(s) for a particular daemon
#
# Inputs: -p pidfile, use the specified pidfile instead of pidof
#         pathname, path to the specified program
#
# Outputs: return 0 - Success, pid's in stdout
#          return 1 - Program is dead, pidfile exists
#          return 2 - Invalid or excessive number of arguments,
#                     warning in stdout
#          return 3 - Program is not running
#
# Dependencies: pidof, echo, head
#
# Todo: Remove dependency on head
#       This replaces getpids
#       Test changes to pidof
#
#**************************************************************************
****
pidofproc()
{
   local pidfile=""
   local lpids=""
   local silent=""
   pidlist=""
   while true
   do
      case "${1}" in
         -p)
            pidfile="${2}"
            shift 2
            ;;

         -s)
            # Added for legacy opperation of getpids
            # eliminates several '> /dev/null'
            silent="1"
            shift 1
            ;;
         -*)
            log_failure_msg "Unknown Option: ${1}"
            return 2
            ;;
         *)
            break
            ;;
      esac
   done

   if [ "${#}" != "1" ]; then
      shift 1
      log_failure_msg "Usage: pidofproc [-s] [-p pidfile] pathname"
      return 2
   fi
```

```
   if [ -n "${pidfile}" ]; then
      if [ ! -r "${pidfile}" ]; then
         return 3 # Program is not running
      fi

      lpids=`head -n 1 ${pidfile}`
      for pid in ${lpids}
      do
         if [ "${pid}" -ne "$$" -a "${pid}" -ne "${PPID}" ]; then
            kill -0 "${pid}" 2>/dev/null &&
            pidlist="${pidlist} ${pid}"
         fi

         if [ "${silent}" != "1" ]; then
            echo "${pidlist}"
         fi

         test -z "${pidlist}" &&
         # Program is dead, pidfile exists
         return 1
         # else
         return 0
      done

   else
      pidlist=`pidof -o $$ -o $PPID -x "$1"`
      if [ "${silent}" != "1" ]; then
         echo "${pidlist}"
      fi

      # Get provide correct running status
      if [ -n "${pidlist}" ]; then
         return 0
      else
         return 3
      fi

   fi

   if [ "$?" != "0" ]; then
      return 3 # Program is not running
   fi
}

#***********************************************************************
****
# Function - loadproc [-f] [-n nicelevel] [-p pidfile] pathname [args]
#
# Purpose: This runs the specified program as a daemon
#
# Inputs: -f, run the program even if it is already running
#         -n nicelevel, specifies a nice level. See nice(1).
#         -p pidfile, uses the specified pidfile
#         pathname, pathname to the specified program
```

```
#         args, arguments to pass to specified program
#
# Outputs: return 0 - Success
#          return 2 - Invalid of excessive number of arguments,
#                     warning in stdout
#          return 4 - Program or service status is unknown
#
# Dependencies: nice, rm
#
# Todo: LSB says this should be called start_daemon
#       LSB does not say that it should call evaluate_retval
#       It checks for PIDFILE, which is deprecated.
#         Will be removed after BLFS 6.0
#       loadproc returns 0 if program is already running, not LSB compliant
#
#**********************************************************************
****
loadproc()
{
   local pidfile=""
   local forcestart=""
   local nicelevel="10"

# This will ensure compatibility with previous LFS Bootscripts
   if [ -n "${PIDFILE}" ]; then
      pidfile="${PIDFILE}"
   fi

  while true
   do
      case "${1}" in
         -f)
            forcestart="1"
            shift 1
            ;;
         -n)
            nicelevel="${2}"
            shift 2
            ;;
         -p)
            pidfile="${2}"
            shift 2
            ;;
         -*)
            log_failure_msg "Unknown Option: ${1}"
            return 2 #invalid or excess argument(s)
            ;;
         *)
            break
            ;;
      esac
   done

   if [ "${#}" = "0" ]; then
      log_failure_msg "Usage: loadproc [-f] [-n nicelevel] [-p pidfile]
```

```
pathname [args]"
      return 2 #invalid or excess argument(s)
   fi

   if [ -z "${forcestart}" ]; then
      if [ -z "${pidfile}" ]; then
         pidofproc -s "${1}"
      else
         pidofproc -s -p "${pidfile}" "${1}"
      fi

      case "${?}" in
         0)
            log_warning_msg "Unable to continue: ${1} is running"
            return 0 # 4
            ;;
         1)
            boot_mesg "Removing stale pid file: ${pidfile}" ${WARNING}
            rm -f "${pidfile}"
            ;;
         3)
            ;;
         *)
            log_failure_msg "Unknown error code from pidofproc: ${?}"
            return 4
            ;;
      esac
   fi

   nice -n "${nicelevel}" "${@}"
   evaluate_retval # This is "Probably" not LSB compliant,
#                          but required to be compatible with older
bootscripts
   return 0
}

#*********************************************************************
****
# Function - killproc  [-p pidfile] pathname [signal]
#
# Purpose:
#
# Inputs: -p pidfile, uses the specified pidfile
#         pathname, pathname to the specified program
#         signal, send this signal to pathname
#
# Outputs: return 0 - Success
#          return 2 - Invalid of excessive number of arguments,
#                     warning in stdout
#          return 4 - Unknown Status
#
# Dependencies: kill, rm
#
# Todo: LSB does not say that it should call evaluate_retval
#       It checks for PIDFILE, which is deprecated.
```

```
#         Will be removed after BLFS 6.0
#
#*********************************************************************
****
killproc()
{
   local pidfile=""
   local killsig=TERM # default signal is SIGTERM
   pidlist=""

   # This will ensure compatibility with previous LFS Bootscripts
   if [ -n "${PIDFILE}" ]; then
      pidfile="${PIDFILE}"
   fi

   while true
   do
      case "${1}" in
         -p)
            pidfile="${2}"
            shift 2
            ;;
         -*)
            log_failure_msg "Unknown Option: ${1}"
            return 2
            ;;
         *)
            break
            ;;
      esac
   done

   if [ "${#}" = "2" ]; then
      killsig="${2}"
   elif [ "${#}" != "1" ]; then
      shift 2
      log_failure_msg "Usage: killproc  [-p pidfile] pathname [signal]"
      return 2
   fi

   # Is the process running?
   if [ -z "${pidfile}" ]; then
      pidofproc -s "${1}"
   else
      pidofproc -s -p "${pidfile}" "${1}"
   fi

   # Remove stale pidfile
   if [ "$?" = 1 ]; then
      boot_mesg "Removing stale pid file: ${pidfile}." ${WARNING}
      rm -f "${pidfile}"
   fi

    # If running, send the signal
    if [ -n "${pidlist}" ]; then
```

```
   for pid in ${pidlist}
   do
      kill -${killsig} ${pid} 2>/dev/null

      # Wait up to 3 seconds, for ${pid} to terminate
      case "${killsig}" in
      TERM|SIGTERM|KILL|SIGKILL)
         # sleep in 1/10ths of seconds and
         # multiply KILLDELAY by 10
         local dtime="${KILLDELAY}0"
         while [ "${dtime}" != "0" ]
         do
            kill -0 ${pid} 2>/dev/null || break
            sleep 0.1
            dtime=$(( ${dtime} - 1))
         done
         # If ${pid} is still running, kill it
         kill -0 ${pid} 2>/dev/null && kill -KILL ${pid} 2>/dev/null
         ;;
      esac
   done

   # Check if the process is still running if we tried to stop it
   case "${killsig}" in
   TERM|SIGTERM|KILL|SIGKILL)
      if [ -z "${pidfile}" ]; then
         pidofproc -s "${1}"
      else
         pidofproc -s -p "${pidfile}" "${1}"
      fi

      # Program was terminated
      if [ "$?" != "0" ]; then
         # Remove the pidfile if necessary
         if [ -f "${pidfile}" ]; then
            rm -f "${pidfile}"
         fi
         echo_ok
         return 0
      else # Program is still running
         echo_failure
         return 4 # Unknown Status
      fi
      ;;
   *)
      # Just see if the kill returned successfully
      evaluate_retval
      ;;
   esac
    else # process not running
   print_status warning not_running
    fi
}
```

```
#***************************************************************************
****
# Function - log_success_msg "message"
#
# Purpose: Print a success message
#
# Inputs: $@ - Message
#
# Outputs: Text output to screen
#
# Dependencies: echo
#
# Todo: logging
#
#***************************************************************************
****
log_success_msg()
{
   ${ECHO} -n -e "${BOOTMESG_PREFIX}${@}"
   ${ECHO} -e "${SET_COL}""${BRACKET}""[""${SUCCESS}""  OK  ""$
{BRACKET}""]""${NORMAL}"

   [ -d /run/var ] || return 0
   ${ECHO} -n -e "${@}  [ OK ]"  >> /run/var/bootlog
   return 0
}

#***************************************************************************
****
# Function - log_failure_msg "message"
#
# Purpose: Print a failure message
#
# Inputs: $@ - Message
#
# Outputs: Text output to screen
#
# Dependencies: echo
#
# Todo: logging
#
#***************************************************************************
****
log_failure_msg() {
   ${ECHO} -n -e "${BOOTMESG_PREFIX}${@}"
   ${ECHO} -e "${SET_COL}""${BRACKET}""[""${FAILURE}"" FAIL ""$
{BRACKET}""]""${NORMAL}"

   [ -d /run/var ] || return 0
   ${ECHO} -e "${@}  [ FAIL ]" >> /run/var/bootlog
   return 0
}

#***************************************************************************
****
```

```
# Function - log_warning_msg "message"
#
# Purpose: print a warning message
#
# Inputs: $@ - Message
#
# Outputs: Text output to screen
#
# Dependencies: echo
#
# Todo: logging
#
#*****************************************************************************
****
log_warning_msg() {
   ${ECHO} -n -e "${BOOTMESG_PREFIX}${@}"
   ${ECHO} -e "${SET_COL}""${BRACKET}""["${WARNING}"" WARN ""$
{BRACKET}""]""${NORMAL}"

   [ -d /run/var ] || return 0
   ${ECHO} -e "${@}  [ WARN ]" >> /run/var/bootlog
   return 0
}

#*****************************************************************************
****
# Function - log_skipped_msg "message"
#
# Purpose: print a message that the script was skipped
#
# Inputs: $@ - Message
#
# Outputs: Text output to screen
#
# Dependencies: echo
#
# Todo: logging
#
#*****************************************************************************
****
log_skipped_msg() {
   ${ECHO} -n -e "${BOOTMESG_PREFIX}${@}"
   ${ECHO} -e "${SET_COL}""${BRACKET}""["${WARNING}"" SKIP ""$
{BRACKET}""]""${NORMAL}"

   [ -d /run/var ] || return 0
   ${ECHO} -e "${@}  [ SKIP ]" >> /run/var/bootlog
   return 0
}

# End boot functions
```

Г.4. /etc/rc.d/init.d/mountvirtfs

```
#!/bin/sh
```

```
#############################################################################
# Begin mountvirtfs
#
# Description : Mount proc, sysfs, and run
#
# Authors     : Gerard Beekmans - gerard@linuxfromscratch.org
#               DJ Lucas - dj@linuxfromscratch.org
# Update      : Bruce Dubbs - bdubbs@linuxfromscratch.org
#
# Version     : LFS 7.0
#
#############################################################################

### BEGIN INIT INFO
# Provides:            mountvirtfs
# Required-Start:
# Should-Start:
# Required-Stop:
# Should-Stop:
# Default-Start:       S
# Default-Stop:
# Short-Description:   Mounts /sys and /proc virtual (kernel) filesystems.
#                      Mounts /run (tmpfs) and /dev (devtmpfs).
# Description:         Mounts /sys and /proc virtual (kernel) filesystems.
#                      Mounts /run (tmpfs) and /dev (devtmpfs).
# X-LFS-Provided-By:   LFS
### END INIT INFO

. /lib/lsb/init-functions

case "${1}" in
   start)
      # Make sure /run/var is available before logging any messages
      if ! mountpoint /run >/dev/null; then
         mount -n /run || failed=1
      fi

      mkdir -p /run/var /run/lock /run/shm
      chmod 1777 /run/shm

      log_info_msg "Mounting virtual file systems: ${INFO}/run"

      if ! mountpoint /proc >/dev/null; then
         log_info_msg2 " ${INFO}/proc"
         mount -n -o nosuid,noexec,nodev /proc || failed=1
      fi

      if ! mountpoint /sys >/dev/null; then
         log_info_msg2 " ${INFO}/sys"
         mount -n -o nosuid,noexec,nodev /sys || failed=1
      fi

      if ! mountpoint /dev >/dev/null; then
         log_info_msg2 " ${INFO}/dev"
         mount -n -o mode=0755,nosuid /dev  || failed=1
```

```
      fi

      # Copy devices that Udev >= 155 doesn't handle to /dev
      cp -a /lib/udev/devices/* /dev

      ln -sfn /run/shm /dev/shm

      (exit ${failed})
      evaluate_retval
      exit $failed
      ;;

   *)
      echo "Usage: ${0} {start}"
      exit 1
      ;;
esac

# End mountvirtfs
```

Г.5. /etc/rc.d/init.d/modules

```
#!/bin/sh
########################################################################
# Begin modules
#
# Description : Module auto-loading script
#
# Authors     : Zack Winkles
#               DJ Lucas - dj@linuxfromscratch.org
# Update      : Bruce Dubbs - bdubbs@linuxfromscratch.org
#
# Version     : LFS 7.0
#
########################################################################

### BEGIN INIT INFO
# Provides:            modules
# Required-Start:      mountvirtfs sysctl
# Should-Start:
# Required-Stop:
# Should-Stop:
# Default-Start:       S
# Default-Stop:
# Short-Description:   Loads required modules.
# Description:         Loads modules listed in /etc/sysconfig/modules.
# X-LFS-Provided-By:   LFS
### END INIT INFO

# Assure that the kernel has module support.
[ -e /proc/ksyms -o -e /proc/modules ] || exit 0

. /lib/lsb/init-functions

case "${1}" in
```

```
   start)
      # Exit if there's no modules file or there are no
      # valid entries
      [ -r /etc/sysconfig/modules ]             || exit 0
      egrep -qv '^($|#)' /etc/sysconfig/modules || exit 0

      log_info_msg "Loading modules:"

      # Only try to load modules if the user has actually given us
      # some modules to load.

      while read module args; do

         # Ignore comments and blank lines.
         case "$module" in
            ""|"#"*) continue ;;
         esac

         # Attempt to load the module, passing any arguments provided.
         modprobe ${module} ${args} >/dev/null

         # Print the module name if successful, otherwise take note.
         if [ $? -eq 0 ]; then
            log_info_msg2 " ${module}"
         else
            failedmod="${failedmod} ${module}"
         fi
      done < /etc/sysconfig/modules

      # Print a message about successfully loaded modules on the correct
line.
      log_success_msg2

      # Print a failure message with a list of any modules that
      # may have failed to load.
      if [ -n "${failedmod}" ]; then
         log_failure_msg "Failed to load modules:${failedmod}"
         exit 1
      fi
      ;;

   *)
      echo "Usage: ${0} {start}"
      exit 1
      ;;
esac

exit 0

# End modules
```

Г.6. /etc/rc.d/init.d/udev

```
#!/bin/sh
########################################################################
```

```
# Begin udev
#
# Description : Udev cold-plugging script
#
# Authors     : Zack Winkles, Alexander E. Patrakov
#               DJ Lucas - dj@linuxfromscratch.org
# Update      : Bruce Dubbs - bdubbs@linuxfromscratch.org
#
# Version     : LFS 7.0
#
########################################################################

### BEGIN INIT INFO
# Provides:            udev $time
# Required-Start:
# Should-Start:        modules
# Required-Stop:
# Should-Stop:
# Default-Start:       S
# Default-Stop:
# Short-Description:   Populates /dev with device nodes.
# Description:         Mounts a tempfs on /dev and starts the udevd daemon.
#                      Device nodes are created as defined by udev.
# X-LFS-Provided-By:   LFS
### END INIT INFO

. /lib/lsb/init-functions

case "${1}" in
   start)
      log_info_msg "Populating /dev with device nodes... "
      if ! grep -q '[[:space:]]sysfs' /proc/mounts; then
         log_failure_msg2
         msg="FAILURE:\n\nUnable to create "
         msg="${msg}devices without a SysFS filesystem\n\n"
         msg="${msg}After you press Enter, this system "
         msg="${msg}will be halted and powered off.\n\n"
         log_info_msg "$msg"
         log_info_msg "Press Enter to continue..."
         wait_for_user
         /etc/rc.d/init.d/halt stop
      fi

      # Udev handles uevents itself, so we don't need to have
      # the kernel call out to any binary in response to them
      echo > /proc/sys/kernel/hotplug

      # Start the udev daemon to continually watch for, and act on,
      # uevents
      /lib/udev/udevd --daemon

      # Now traverse /sys in order to "coldplug" devices that have
      # already been discovered
      /sbin/udevadm trigger --action=add --type=subsystems
      /sbin/udevadm trigger --action=add --type=devices
```

```
      # Now wait for udevd to process the uevents we triggered
      /sbin/udevadm settle

      # 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
```

Г.7. /etc/rc.d/init.d/swap

```
#!/bin/sh
########################################################################
# Begin swap
#
# Description : Swap Control Script
#
# Authors     : Gerard Beekmans - gerard@linuxfromscratch.org
#               DJ Lucas - dj@linuxfromscratch.org
# Update      : Bruce Dubbs - bdubbs@linuxfromscratch.org
#
# Version     : LFS 7.0
#
########################################################################

### BEGIN INIT INFO
# Provides:            swap
# Required-Start:      udev
# Should-Start:        modules
# Required-Stop:       localnet
# Should-Stop:
# Default-Start:       S
# Default-Stop:        0 6
# Short-Description:   Mounts and unmounts swap partitions.
# Description:         Mounts and unmounts swap partitions defined in
#                      /etc/fstab.
# X-LFS-Provided-By:   LFS
### END INIT INFO

. /lib/lsb/init-functions

case "${1}" in
   start)
```

```
      log_info_msg "Activating all swap files/partitions..."
      swapon -a
      evaluate_retval
      ;;

   stop)
      log_info_msg "Deactivating all swap files/partitions..."
      swapoff -a
      evaluate_retval
      ;;

   restart)
      ${0} stop
      sleep 1
      ${0} start
      ;;

   status)
      log_success_msg "Retrieving swap status."
      swapon -s
      ;;

   *)
      echo "Usage: ${0} {start|stop|restart|status}"
      exit 1
      ;;
esac

exit 0

# End swap
```

Г.8. /etc/rc.d/init.d/setclock

```
#!/bin/sh
########################################################################
# Begin setclock
#
# Description : Setting Linux Clock
#
# Authors     : Gerard Beekmans - gerard@linuxfromscratch.org
#               DJ Lucas - dj@linuxfromscratch.org
# Update      : Bruce Dubbs - bdubbs@linuxfromscratch.org
#
# Version     : LFS 7.0
#
########################################################################

### BEGIN INIT INFO
# Provides:
# Required-Start:
# Should-Start:        modules
# Required-Stop:
# Should-Stop:         $syslog
# Default-Start:       S
```

```
# Default-Stop:
# Short-Description:   Stores and restores time from the hardware clock
# Description:         On boot, system time is obtained from hwclock.  The
#                      hardware clock can also be set on shutdown.
# X-LFS-Provided-By:   LFS BLFS
### END INIT INFO

. /lib/lsb/init-functions

[ -r /etc/sysconfig/clock ] && . /etc/sysconfig/clock


case "${UTC}" in
   yes|true|1)
      CLOCKPARAMS="${CLOCKPARAMS} --utc"
      ;;

   no|false|0)
      CLOCKPARAMS="${CLOCKPARAMS} --localtime"
      ;;

esac

case ${1} in
   start)
      hwclock --hctosys ${CLOCKPARAMS} >/dev/null
      ;;

   stop)
      log_info_msg "Setting hardware clock..."
      hwclock --systohc ${CLOCKPARAMS} >/dev/null
      evaluate_retval
      ;;

   *)
      echo "Usage: ${0} {start|stop}"
      exit 1
      ;;

esac

exit 0
```

Г.9. /etc/rc.d/init.d/checkfs

```
#!/bin/sh
########################################################################
# Begin checkfs
#
# Description : File System Check
#
# Authors     : Gerard Beekmans - gerard@linuxfromscratch.org
#               A. Luebke - luebke@users.sourceforge.net
#               DJ Lucas - dj@linuxfromscratch.org
# Update      : Bruce Dubbs - bdubbs@linuxfromscratch.org
```

```
#
# Version     : LFS 7.0
#
# Based on checkfs script from LFS-3.1 and earlier.
#
# From man fsck
# 0    - No errors
# 1    - File system errors corrected
# 2    - System should be rebooted
# 4    - File system errors left uncorrected
# 8    - Operational error
# 16   - Usage or syntax error
# 32   - Fsck canceled by user request
# 128  - Shared library error
#
########################################################################

### BEGIN INIT INFO
# Provides:            checkfs
# Required-Start:      udev swap $time
# Should-Start:
# Required-Stop:
# Should-Stop:
# Default-Start:       S
# Default-Stop:
# Short-Description:   Checks local filesystems before mounting.
# Description:         Checks local filesystmes before mounting.
# X-LFS-Provided-By:   LFS
### END INIT INFO

. /lib/lsb/init-functions

case "${1}" in
   start)
      if [ -f /fastboot ]; then
         msg="/fastboot found, will omit "
         msg="${msg} file system checks as requested.\n"
         log_info_msg "${msg}"
         exit 0
      fi

      log_info_msg "Mounting root file system in read-only mode... "
      mount -n -o remount,ro / >/dev/null

      if [ ${?} != 0 ]; then
         log_failure_msg2
         msg="\n\nCannot check root "
         msg="${msg}filesystem because it could not be mounted "
         msg="${msg}in read-only mode.\n\n"
         msg="${msg}After you press Enter, this system will be "
         msg="${msg}halted and powered off.\n\n"
         log_failure_msg "${msg}"

         log_info_msg "Press Enter to continue..."
         wait_for_user
```

```
      /etc/rc.d/init.d/halt stop
   else
      log_success_msg2
   fi

   if [ -f /forcefsck ]; then
      msg="\n/forcefsck found, forcing file"
      msg="${msg} system checks as requested."
      log_success_msg "$msg"
      options="-f"
   else
      options=""
   fi

   log_info_msg "Checking file systems..."
   # Note: -a option used to be -p; but this fails e.g. on fsck.minix
   fsck ${options} -a -A -C -T >/dev/null
   error_value=${?}

   if [ "${error_value}" = 0 ]; then
      log_success_msg2
   fi

   if [ "${error_value}" = 1 ]; then
      msg="\nWARNING:\n\nFile system errors "
      msg="${msg}were found and have been corrected.\n"
      msg="${msg}You may want to double-check that "
      msg="${msg}everything was fixed properly."
      log_warning_msg "$msg"
   fi

   if [ "${error_value}" = 2 -o "${error_value}" = 3 ]; then
      msg="\nWARNING:\n\nFile system errors "
      msg="${msg}were found and have been been "
      msg="${msg}corrected, but the nature of the "
      msg="${msg}errors require this system to be rebooted.\n\n"
      msg="${msg}After you press enter, "
      msg="${msg}this system will be rebooted\n\n"
      log_failure_msg "$msg"

      log_info_msg "Press Enter to continue..."
      wait_for_user
      reboot -f
   fi

   if [ "${error_value}" -gt 3 -a "${error_value}" -lt 16 ]; then
      msg="\nFAILURE:\n\nFile system errors "
      msg="${msg}were encountered that could not be "
      msg="${msg}fixed automatically.  This system "
      msg="${msg}cannot continue to boot and will "
      msg="${msg}therefore be halted until those "
      msg="${msg}errors are fixed manually by a "
      msg="${msg}System Administrator.\n\n"
      msg="${msg}After you press Enter, this system will be "
      msg="${msg}halted and powered off.\n\n"
```

```
         log_failure_msg "$msg"

         log_info_msg "Press Enter to continue..."
         wait_for_user
         /etc/rc.d/init.d/halt stop
      fi

      if [ "${error_value}" -ge 16 ]; then
         msg="\nFAILURE:\n\nUnexpected Failure "
         msg="${msg}running fsck.  Exited with error "
         msg="${msg} code: ${error_value}."
         log_failure_msg $msg
         exit ${error_value}
      fi

      exit 0
      ;;
   *)
      echo "Usage: ${0} {start}"
      exit 1
      ;;
esac

# End checkfs
```

Г.10. /etc/rc.d/init.d/mountfs

```
#!/bin/sh
########################################################################
# Begin mountfs
#
# Description : File System Mount Script
#
# Authors     : Gerard Beekmans - gerard@linuxfromscratch.org
#               DJ Lucas - dj@linuxfromscratch.org
# Update      : Bruce Dubbs - bdubbs@linuxfromscratch.org
#
# Version     : LFS 7.0
#
########################################################################

### BEGIN INIT INFO
# Provides:            $local_fs
# Required-Start:      udev checkfs
# Should-Start:
# Required-Stop:       swap
# Should-Stop:
# Default-Start:       S
# Default-Stop:        0 6
# Short-Description:   Mounts/unmounts local filesystems defined in
/etc/fstab.
# Description:         Remounts root filesystem read/write and mounts all
#                      remaining local filesystems defined in /etc/fstab on
#                      start.  Remounts root filesystem read-only and
unmounts
```

```
#                       remaining filesystems on stop.
# X-LFS-Provided-By:    LFS
### END INIT INFO

. /lib/lsb/init-functions

case "${1}" in
   start)
      log_info_msg "Remounting root file system in read-write mode..."
      mount -n -o remount,rw / >/dev/null
      evaluate_retval

      # Remove fsck-related file system watermarks.
      rm -f /fastboot /forcefsck

      log_info_msg "Recording existing mounts in /etc/mtab..."
      > /etc/mtab

      mount -f /     || failed=1
      mount -f /proc || failed=1
      mount -f /sys  || failed=1
      mount -f /run  || failed=1
      mount -f /dev  || failed=1
      (exit ${failed})
      evaluate_retval

      # This will mount all filesystems that do not have _netdev in
      # their option list.  _netdev denotes a network filesystem.

      log_info_msg "Mounting remaining file systems..."
      mount -a -O no_netdev >/dev/null
      evaluate_retval
      exit $failed
      ;;

   stop)
      # Don't unmount tmpfs like /run
      log_info_msg "Unmounting all other currently mounted file systems..."
      umount -a -d -r -t notmpfs,nosysfs,nodevtmpfs,noproc >/dev/null
      evaluate_retval

      # Make all LVM volume groups unavailable, if appropriate
      # This fails if swap or / are on an LVM partition
      #if [ -x /sbin/vgchange ]; then /sbin/vgchange -an > /dev/null; fi
      ;;

   *)
      echo "Usage: ${0} {start|stop}"
      exit 1
      ;;
esac

# End mountfs
```

Г.11. /etc/rc.d/init.d/udev_retry

```
#!/bin/sh
########################################################################
# Begin udev_retry
#
# Description : Udev cold-plugging script (retry)
#
# Authors     : Alexander E. Patrakov
#               DJ Lucas - dj@linuxfromscratch.org
# Update      : Bruce Dubbs - bdubbs@linuxfromscratch.org
#               Bryan Kadzban -
#
# Version     : LFS 7.0
#
########################################################################

### BEGIN INIT INFO
# Provides:            udev_retry
# Required-Start:      udev
# Should-Start:        $local_fs
# Required-Stop:
# Should-Stop:
# Default-Start:       S
# Default-Stop:
# Short-Description:   Replays failed uevents and creates additional
devices.
# Description:         Replays any failed uevents that were skipped due to
#                      slow hardware initialization, and creates those
needed
#                      device nodes
# X-LFS-Provided-By:   LFS
### END INIT INFO

. /lib/lsb/init-functions

case "${1}" in
   start)
      log_info_msg "Retrying failed uevents, if any..."

      # As of udev-186, the --run option is no longer valid
      #rundir=$(/sbin/udevadm info --run)
      rundir=/run/udev
      # From Debian: "copy the rules generated before / was mounted
      # read-write":

      for file in ${rundir}/tmp-rules--*; do
         dest=${file##*tmp-rules--}
         [ "$dest" = '*' ] && break
         cat $file >> /etc/udev/rules.d/$dest
         rm -f $file
      done

      # Re-trigger the uevents that may have failed,
      # in hope they will succeed now
      /bin/sed -e 's/#.*$//' /etc/sysconfig/udev_retry | /bin/grep -v '^$' |
\
```

```
      while read line ; do
         for subsystem in $line ; do
            /sbin/udevadm trigger --subsystem-match=$subsystem --action=add
         done
      done

      # Now wait for udevd to process the uevents we triggered
      /sbin/udevadm settle
      evaluate_retval
      ;;

   *)
      echo "Usage ${0} {start}"
      exit 1
      ;;
esac

exit 0

# End udev_retry
```

Г.12. /etc/rc.d/init.d/cleanfs

```
#!/bin/sh
########################################################################
# Begin cleanfs
#
# Description : Clean file system
#
# Authors     : Gerard Beekmans - gerard@linuxfromscratch.org
#               DJ Lucas - dj@linuxfromscratch.org
# Update      : Bruce Dubbs - bdubbs@linuxfromscratch.org
#
# Version     : LFS 7.0
#
########################################################################

### BEGIN INIT INFO
# Provides:            cleanfs
# Required-Start:      $local_fs
# Should-Start:
# Required-Stop:
# Should-Stop:
# Default-Start:       S
# Default-Stop:
# Short-Description:   Cleans temporary directories early in the boot
process.
# Description:         Cleans temporary directories /var/run, /var/lock, and
#                      optionally, /tmp.  cleanfs also creates /var/run/utmp
#                      and any files defined in /etc/sysconfig/createfiles.
# X-LFS-Provided-By:   LFS
### END INIT INFO

. /lib/lsb/init-functions
```

```
# Function to create files/directory on boot.
create_files()
{
   # Input to file descriptor 9 and output to stdin (redirection)
   exec 9>&0 < /etc/sysconfig/createfiles

   while read name type perm usr grp dtype maj min junk
   do
      # Ignore comments and blank lines.
      case "${name}" in
         ""|\#*) continue ;;
      esac

      # Ignore existing files.
      if [ ! -e "${name}" ]; then
         # Create stuff based on its type.
         case "${type}" in
            dir)
               mkdir "${name}"
               ;;
            file)
               :> "${name}"
               ;;
            dev)
               case "${dtype}" in
                  char)
                     mknod "${name}" c ${maj} ${min}
                     ;;
                  block)
                     mknod "${name}" b ${maj} ${min}
                     ;;
                  pipe)
                     mknod "${name}" p
                     ;;
                  *)
                     log_warning_msg "\nUnknown device type: ${dtype}"
                     ;;
               esac
               ;;
            *)
               log_warning_msg "\nUnknown type: ${type}"
               continue
               ;;
         esac

         # Set up the permissions, too.
         chown ${usr}:${grp} "${name}"
         chmod ${perm} "${name}"
      fi
   done

   # Close file descriptor 9 (end redirection)
   exec 0>&9 9>&-
   return 0
}
```

```
case "${1}" in
   start)
      log_info_msg "Cleaning file systems:"

      if [ "${SKIPTMPCLEAN}" = "" ]; then
         log_info_msg2 " /tmp"
         cd /tmp &&
         find . -xdev -mindepth 1 ! -name lost+found -delete || failed=1
      fi

      > /var/run/utmp

      if grep -q '^utmp:' /etc/group ; then
         chmod 664 /var/run/utmp
         chgrp utmp /var/run/utmp
      fi

      (exit ${failed})
      evaluate_retval

      if egrep -qv '^(#|$)' /etc/sysconfig/createfiles 2>/dev/null; then
         log_info_msg "Creating files and directories... "
         create_files      # Always returns 0
         evaluate_retval
      fi

      exit $failed
      ;;
   *)
      echo "Usage: ${0} {start}"
      exit 1
      ;;
esac

# End cleanfs
```

Г.13. /etc/rc.d/init.d/console

```
#!/bin/sh
########################################################################
# Begin console
#
# Description : Sets keymap and screen font
#
# Authors     : Gerard Beekmans - gerard@linuxfromscratch.org
#               Alexander E. Patrakov
#               DJ Lucas - dj@linuxfromscratch.org
# Update      : Bruce Dubbs - bdubbs@linuxfromscratch.org
#
# Version     : LFS 7.0
#
########################################################################

### BEGIN INIT INFO
```

```
# Provides:            console
# Required-Start:
# Should-Start:        $local_fs
# Required-Stop:
# Should-Stop:
# Default-Start:       S
# Default-Stop:
# Short-Description:   Sets up a localised console.
# Description:         Sets up fonts and language settings for the user's
#                      local as defined by /etc/sysconfig/console.
# X-LFS-Provided-By:   LFS
### END INIT INFO

. /lib/lsb/init-functions

# Native English speakers probably don't have /etc/sysconfig/console at all
[ -r /etc/sysconfig/console ] && . /etc/sysconfig/console

function is_true()
{
   [ "$1" = "1" ] || [ "$1" = "yes" ] || [ "$1" = "true" ]
}

failed=0

case "${1}" in
   start)
      # See if we need to do anything
      if [ -z "${KEYMAP}"         ] && [ -z "${KEYMAP_CORRECTIONS}" ] &&
         [ -z "${FONT}"           ] && [ -z "${LEGACY_CHARSET}"     ] &&
         ! is_true "${UNICODE}"; then
         exit 0
      fi

      # There should be no bogus failures below this line!
      log_info_msg "Setting up Linux console..."

      # Figure out if a framebuffer console is used
      [ -d /sys/class/graphics/fb0 ] && use_fb=1 || use_fb=0

      # Figure out the command to set the console into the
      # desired mode
      is_true "${UNICODE}" &&
         MODE_COMMAND="echo -en '\033%G' && kbd_mode -u" ||
         MODE_COMMAND="echo -en '\033%@\033(K' && kbd_mode -a"

      # On framebuffer consoles, font has to be set for each vt in
      # UTF-8 mode. This doesn't hurt in non-UTF-8 mode also.

      ! is_true "${use_fb}" || [ -z "${FONT}" ] ||
         MODE_COMMAND="${MODE_COMMAND} && setfont ${FONT}"

      # Apply that command to all consoles mentioned in
      # /etc/inittab. Important: in the UTF-8 mode this should
      # happen before setfont, otherwise a kernel bug will
```

```
      # show up and the unicode map of the font will not be
      # used.

      for TTY in `grep '^[^#].*respawn:/sbin/agetty' /etc/inittab |
         grep -o '\btty[[:digit:]]*\b'`
      do
         openvt -f -w -c ${TTY#tty} -- \
            /bin/sh -c "${MODE_COMMAND}" || failed=1
      done

      # Set the font (if not already set above) and the keymap
      [ "${use_fb}" == "1" ] || [ -z "${FONT}" ] || setfont $FONT ||
failed=1

      [ -z "${KEYMAP}" ] ||
         loadkeys ${KEYMAP} >/dev/null 2>&1 ||
         failed=1

      [ -z "${KEYMAP_CORRECTIONS}" ] ||
         loadkeys ${KEYMAP_CORRECTIONS} >/dev/null 2>&1 ||
         failed=1

      # Convert the keymap from $LEGACY_CHARSET to UTF-8
      [ -z "$LEGACY_CHARSET" ] ||
         dumpkeys -c "$LEGACY_CHARSET" | loadkeys -u >/dev/null 2>&1 ||
         failed=1

      # If any of the commands above failed, the trap at the
      # top would set $failed to 1
      ( exit $failed )
      evaluate_retval

      exit $failed
      ;;

   *)
      echo "Usage:  ${0} {start}"
      exit 1
      ;;
esac

# End console
```

Г.14. /etc/rc.d/init.d/localnet

```
#!/bin/sh
########################################################################
# Begin localnet
#
# Description : Loopback device
#
# Authors     : Gerard Beekmans  - gerard@linuxfromscratch.org
#               DJ Lucas - dj@linuxfromscratch.org
# Update      : Bruce Dubbs - bdubbs@linuxfromscratch.org
#
```

```
# Version     : LFS 7.0
#
########################################################################

### BEGIN INIT INFO
# Provides:            localnet
# Required-Start:      $local_fs
# Should-Start:
# Required-Stop:
# Should-Stop:
# Default-Start:       S
# Default-Stop:        0 6
# Short-Description:   Starts the local network.
# Description:         Sets the hostname of the machine and starts the
#                      loopback interface.
# X-LFS-Provided-By:   LFS
### END INIT INFO

. /lib/lsb/init-functions
[ -r /etc/sysconfig/network ] && . /etc/sysconfig/network

case "${1}" in
   start)
      log_info_msg "Bringing up the loopback interface..."
      ip addr add 127.0.0.1/8 label lo dev lo
      ip link set lo up
      evaluate_retval

      log_info_msg "Setting hostname to ${HOSTNAME}..."
      hostname ${HOSTNAME}
      evaluate_retval
      ;;

   stop)
      log_info_msg "Bringing down the loopback interface..."
      ip link set lo down
      evaluate_retval
      ;;

   restart)
      ${0} stop
      sleep 1
      ${0} start
      ;;

   status)
      echo "Hostname is: $(hostname)"
      ip link show lo
      ;;

   *)
      echo "Usage: ${0} {start|stop|restart|status}"
      exit 1
      ;;
esac
```

```
exit 0

# End localnet
```

Г.15. /etc/rc.d/init.d/sysctl

```
#!/bin/sh
########################################################################
# Begin sysctl
#
# Description : File uses /etc/sysctl.conf to set kernel runtime
#               parameters
#
# Authors     : Nathan Coulson (nathan@linuxfromscratch.org)
#               Matthew Burgress (matthew@linuxfromscratch.org)
#               DJ Lucas - dj@linuxfromscratch.org
# Update      : Bruce Dubbs - bdubbs@linuxfromscratch.org
#
# Version     : LFS 7.0
#
########################################################################

### BEGIN INIT INFO
# Provides:            sysctl
# Required-Start:      mountvirtfs
# Should-Start:
# Required-Stop:
# Should-Stop:
# Default-Start:       S
# Default-Stop:
# Short-Description:   Makes changes to the proc filesystem
# Description:         Makes changes to the proc filesystem as defined in
#                      /etc/sysctl.conf.  See 'man sysctl(8)'.
# X-LFS-Provided-By:   LFS
### END INIT INFO

. /lib/lsb/init-functions

case "${1}" in
   start)
      if [ -f "/etc/sysctl.conf" ]; then
         log_info_msg "Setting kernel runtime parameters..."
         sysctl -q -p
         evaluate_retval
      fi
      ;;

   status)
      sysctl -a
      ;;

   *)
      echo "Usage: ${0} {start|status}"
      exit 1
```

```
      ;;
esac

exit 0

# End sysctl
```

Г.16. /etc/rc.d/init.d/sysklogd

```
#!/bin/sh
########################################################################
# Begin sysklogd
#
# Description : Sysklogd loader
#
# Authors     : Gerard Beekmans - gerard@linuxfromscratch.org
#               DJ Lucas - dj@linuxfromscratch.org
# Update      : Bruce Dubbs - bdubbs@linuxfromscratch.org
#
# Version     : LFS 7.0
#
########################################################################

### BEGIN INIT INFO
# Provides:            $syslog
# Required-Start:      localnet
# Should-Start:
# Required-Stop:       $local_fs sendsignals
# Should-Stop:
# Default-Start:       2 3 4 5
# Default-Stop:        0 1 6
# Short-Description:   Starts kernel and system log daemons.
# Description:         Starts kernel and system log daemons.
#                      /etc/fstab.
# X-LFS-Provided-By:   LFS
### END INIT INFO

. /lib/lsb/init-functions

case "${1}" in
   start)
      log_info_msg "Starting system log daemon..."
      parms=${SYSKLOGD_PARMS-'-m 0'}
      start_daemon /sbin/syslogd $parms
      evaluate_retval

      log_info_msg "Starting kernel log daemon..."
      start_daemon /sbin/klogd
      evaluate_retval
      ;;

   stop)
      log_info_msg "Stopping kernel log daemon..."
      killproc /sbin/klogd
      evaluate_retval
```

```
      log_info_msg "Stopping system log daemon..."
      killproc /sbin/syslogd
      evaluate_retval
      ;;

   reload)
      log_info_msg "Reloading system log daemon config file..."
      pid=`pidofproc syslogd`
      kill -HUP "${pid}"
      evaluate_retval
      ;;

   restart)
      ${0} stop
      sleep 1
      ${0} start
      ;;

   status)
      statusproc /sbin/syslogd
      statusproc klogd
      ;;

   *)
      echo "Usage: ${0} {start|stop|reload|restart|status}"
      exit 1
      ;;
esac

exit 0

# End sysklogd
```

Г.17. /etc/rc.d/init.d/network

```
#!/bin/sh
########################################################################
# Begin network
#
# Description : Network Control Script
#
# Authors     : Gerard Beekmans - gerard@linuxfromscratch.org
#               Nathan Coulson - nathan@linuxfromscratch.org
#               Kevin P. Fleming - kpfleming@linuxfromscratch.org
#               DJ Lucas - dj@linuxfromscratch.org
# Update      : Bruce Dubbs - bdubbs@linuxfromscratch.org
#
# Version     : LFS 7.0
#
########################################################################

### BEGIN INIT INFO
# Provides:            $network
# Required-Start:      $local_fs swap localnet
```

```
# Should-Start:         $syslog
# Required-Stop:        $local_fs swap localnet
# Should-Stop:          $syslog
# Default-Start:        3 4 5
# Default-Stop:         0 1 2 6
# Short-Description:    Starts and configures network interfaces.
# Description:          Starts and configures network interfaces.
# X-LFS-Provided-By:    LFS
### END INIT INFO

case "${1}" in
   start)
      # Start all network interfaces
      for file in /etc/sysconfig/ifconfig.*
      do
         interface=${file##*/ifconfig.}

         # Skip if $file is * (because nothing was found)
         if [ "${interface}" = "*" ]
         then
            continue
         fi

         /sbin/ifup ${interface}
      done
      ;;

   stop)
      # Reverse list
      net_files=""
      for file in  /etc/sysconfig/ifconfig.*
      do
         net_files="${file} ${net_files}"
      done

      # Stop all network interfaces
      for file in ${net_files}
      do
         interface=${file##*/ifconfig.}

         # Skip if $file is * (because nothing was found)
         if [ "${interface}" = "*" ]
         then
            continue
         fi

         /sbin/ifdown ${interface}
      done
      ;;

   restart)
      ${0} stop
      sleep 1
      ${0} start
      ;;
```

```
   *)
      echo "Usage: ${0} {start|stop|restart}"
      exit 1
      ;;
esac

exit 0

# End network
```

Г.18. /etc/rc.d/init.d/sendsignals

```
#!/bin/sh
########################################################################
# Begin sendsignals
#
# Description : Sendsignals Script
#
# Authors     : Gerard Beekmans - gerard@linuxfromscratch.org
#               DJ Lucas - dj@linuxfromscratch.org
# Update      : Bruce Dubbs - bdubbs@linuxfromscratch.org
#
# Version     : LFS 7.0
#
########################################################################

### BEGIN INIT INFO
# Provides:            sendsignals
# Required-Start:
# Should-Start:
# Required-Stop:       $local_fs swap localnet
# Should-Stop:
# Default-Start:
# Default-Stop:        0 6
# Short-Description:   Attempts to kill remaining processes.
# Description:         Attempts to kill remaining processes.
# X-LFS-Provided-By:   LFS
### END INIT INFO

. /lib/lsb/init-functions

case "${1}" in
   stop)
      log_info_msg "Sending all processes the TERM signal..."
      killall5 -15
      error_value=${?}

      sleep ${KILLDELAY}

      if [ "${error_value}" = 0 -o "${error_value}" = 2 ]; then
         log_success_msg
      else
         log_failure_msg
      fi
```

```
      log_info_msg "Sending all processes the KILL signal..."
      killall5 -9
      error_value=${?}

      sleep ${KILLDELAY}

      if [ "${error_value}" = 0 -o "${error_value}" = 2 ]; then
         log_success_msg
      else
         log_failure_msg
      fi
      ;;

   *)
      echo "Usage: ${0} {stop}"
      exit 1
      ;;

esac

exit 0

# End sendsignals
```

Г.19. /etc/rc.d/init.d/reboot

```
#!/bin/sh
########################################################################
# Begin reboot
#
# Description : Reboot Scripts
#
# Authors     : Gerard Beekmans - gerard@linuxfromscratch.org
#               DJ Lucas - dj@linuxfromscratch.org
# Update      : Bruce Dubbs - bdubbs@linuxfromscratch.org
#
# Version     : LFS 7.0
#
########################################################################

### BEGIN INIT INFO
# Provides:            reboot
# Required-Start:
# Should-Start:
# Required-Stop:
# Should-Stop:
# Default-Start:       6
# Default-Stop:
# Short-Description:   Reboots the system.
# Description:         Reboots the System.
# X-LFS-Provided-By:   LFS
### END INIT INFO

. /lib/lsb/init-functions
```

```
case "${1}" in
   stop)
      log_info_msg "Restarting system..."
      reboot -d -f -i
      ;;

   *)
      echo "Usage: ${0} {stop}"
      exit 1
      ;;

esac

# End reboot
```

Г.20. /etc/rc.d/init.d/halt

```
#!/bin/sh
########################################################################
# Begin halt
#
# Description : Halt Script
#
# Authors     : Gerard Beekmans - gerard@linuxfromscratch.org
#               DJ Lucas - dj@linuxfromscratch.org
# Update      : Bruce Dubbs - bdubbs@linuxfromscratch.org
#
# Version     : LFS 7.0
#
########################################################################

### BEGIN INIT INFO
# Provides:            halt
# Required-Start:
# Should-Start:
# Required-Stop:
# Should-Stop:
# Default-Start:       0
# Default-Stop:
# Short-Description:   Halts the system.
# Description:         Halts the System.
# X-LFS-Provided-By:   LFS
### END INIT INFO

case "${1}" in
   stop)
      halt -d -f -i -p
      ;;

   *)
      echo "Usage: {stop}"
      exit 1
      ;;
esac
```

```
# End halt
```

Г.21. /etc/rc.d/init.d/template

```
#!/bin/sh
########################################################################
# Begin scriptname
#
# Description :
#
# Authors     :
#
# Version     : LFS x.x
#
# Notes       :
#
########################################################################

### BEGIN INIT INFO
# Provides:            template
# Required-Start:
# Should-Start:
# Required-Stop:
# Should-Stop:
# Default-Start:
# Default-Stop:
# Short-Description:
# Description:
# X-LFS-Provided-By:
### END INIT INFO

. /lib/lsb/init-functions

case "${1}" in
   start)
      log_info_msg "Starting..."
      start_daemon fully_qualified_path
      ;;

   stop)
      log_info_msg "Stopping..."
      killproc fully_qualified_path
      ;;

   restart)
      ${0} stop
      sleep 1
      ${0} start
      ;;

   *)
      echo "Usage: ${0} {start|stop|restart}"
      exit 1
      ;;
```

```
esac

exit 0

# End scriptname
```

Г.22. /etc/sysconfig/modules

```
    ########################################################################
# Begin /etc/sysconfig/modules
#
# Description : Module auto-loading configuration
#
# Authors     :
#
# Version     : 00.00
#
# Notes       : The syntax of this file is as follows:
#          <module> [<arg1> <arg2> ...]
#
# Each module should be on it's own line, and any options that you want
# passed to the module should follow it.  The line deliminator is either
# a space or a tab.
########################################################################

# End /etc/sysconfig/modules
```

Г.23. /etc/sysconfig/createfiles

```
########################################################################
# Begin /etc/sysconfig/createfiles
#
# Description : Createfiles script config file
#
# Authors     :
#
# Version     : 00.00
#
# Notes       : The syntax of this file is as follows:
#         if type is equal to "file" or "dir"
#          <filename> <type> <permissions> <user> <group>
#         if type is equal to "dev"
#          <filename> <type> <permissions> <user> <group> <devtype>
#             <major> <minor>
#
#         <filename> is the name of the file which is to be created
#         <type> is either file, dir, or dev.
#               file creates a new file
#               dir creates a new directory
#               dev creates a new device
#         <devtype> is either block, char or pipe
#               block creates a block device
#               char creates a character deivce
#               pipe creates a pipe, this will ignore the <major> and
#           <minor> fields
#         <major> and <minor> are the major and minor numbers used for
```

```
#     the device.
#########################################################################

# End /etc/sysconfig/createfiles
```

Г.24. /etc/sysconfig/udev-retry

```
 ########################################################################
# Begin /etc/sysconfig/udev_retry
#
# Description : udev_retry script configuration
#
# Authors     :
#
# Version     : 00.00
#
# Notes       : Each subsystem that may need to be re-triggered after
mountfs
#               runs should be listed in this file.  Probable subsystems to
be
#               listed here are rtc (due to /var/lib/hwclock/adjtime) and
sound
#               (due to both /var/lib/alsa/asound.state and
/usr/sbin/alsactl).
#               Entries are whitespace-separated.
########################################################################

rtc

# End /etc/sysconfig/udev_retry
```

Г.25. /sbin/ifup

```
#!/bin/sh
########################################################################
# Begin /sbin/ifup
#
# Description : Interface Up
#
# Authors     : Nathan Coulson - nathan@linuxfromscratch.org
#               Kevin P. Fleming - kpfleming@linuxfromscratch.org
# Update      : Bruce Dubbs - bdubbs@linuxfromscratch.org
#
# Version     : LFS 7.2
#
# Notes       : The IFCONFIG variable is passed to the SERVICE script
#               in the /lib/services directory, to indicate what file the
#               service should source to get interface specifications.
#
########################################################################

up()
{
  if ip link show $1 > /dev/null 2>&1; then
     link_status=`ip link show $1`
```

```
     if [ -n "${link_status}" ]; then
        if ! echo "${link_status}" | grep -q UP; then
           ip link set $1 up
        fi
     fi

  else
     log_failure_msg "\nInterface ${IFACE} doesn't exist."
     exit 1
  fi
}

RELEASE="7.2"

USAGE="Usage: $0 [ -hV ] [--help] [--version] interface"
VERSTR="LFS ifup, version ${RELEASE}"

while [ $# -gt 0 ]; do
   case "$1" in
      --help | -h)     help="y"; break ;;

      --version | -V)  echo "${VERSTR}"; exit 0 ;;

      -*)              echo "ifup: ${1}: invalid option" >&2
                       echo "${USAGE}" >& 2
                       exit 2 ;;

      *)               break ;;
   esac
done

if [ -n "$help" ]; then
   echo "${VERSTR}"
   echo "${USAGE}"
   echo
   cat << HERE_EOF
ifup is used to bring up a network interface.  The interface
parameter, e.g. eth0 or eth0:2, must match the trailing part of the
interface specifications file, e.g. /etc/sysconfig/ifconfig.eth0:2.

HERE_EOF
   exit 0
fi

file=/etc/sysconfig/ifconfig.${1}

# Skip backup files
[ "${file}" = "${file%""~""}" ] || exit 0

. /lib/lsb/init-functions

log_info_msg "Bringing up the ${1} interface... "

if [ ! -r "${file}" ]; then
   log_failure_msg2 "${file} is missing or cannot be accessed."
```

```
   exit 1
fi

. $file

if [ "$IFACE" = "" ]; then
   log_failure_msg2 "${file} does not define an interface [IFACE]."
   exit 1
fi

# Do not process this service if started by boot, and ONBOOT
# is not set to yes
if [ "${IN_BOOT}" = "1" -a "${ONBOOT}" != "yes" ]; then
   log_info_msg2 "skipped"
   exit 0
fi

for S in ${SERVICE}; do
  if [ ! -x "/lib/services/${S}" ]; then
    MSG="\nUnable to process ${file}.  Either "
    MSG="${MSG}the SERVICE '${S} was not present "
    MSG="${MSG}or cannot be executed."
    log_failure_msg "$MSG"
    exit 1
  fi
done

# Create/configure the interface
for S in ${SERVICE}; do
  IFCONFIG=${file} /lib/services/${S} ${IFACE} up
done

# Bring up the interface and any components
for I in $IFACE $INTERFACE_COMPONENTS; do up $I; done

# Set MTU if requested. Check if MTU has a "good" value.
if test -n "${MTU}"; then
   if [[ ${MTU} =~ ^[0-9]+$ ]] && [[ $MTU -ge 68 ]] ; then
      for I in $IFACE $INTERFACE_COMPONENTS; do
         ip link set dev $I mtu $MTU;
      done
   else
      log_info_msg2 "Invalid MTU $MTU"
   fi
fi

# Set the route default gateway if requested
if [ -n "${GATEWAY}" ]; then
   if ip route | grep -q default; then
      log_warning_msg "\nGateway already setup; skipping."
   else
      log_info_msg "Setting up default gateway..."
      ip route add default via ${GATEWAY} dev ${IFACE}
      evaluate_retval
   fi
```

```
fi

# End /sbin/ifup
```

Г.26. /sbin/ifdown

```
#!/bin/bash
########################################################################
# Begin /sbin/ifdown
#
# Description : Interface Down
#
# Authors     : Nathan Coulson - nathan@linuxfromscratch.org
#               Kevin P. Fleming - kpfleming@linuxfromscratch.org
# Update      : Bruce Dubbs - bdubbs@linuxfromscratch.org
#
# Version     : LFS 7.0
#
# Notes       : the IFCONFIG variable is passed to the scripts found
#               in the /lib/services directory, to indicate what file the
#               service should source to get interface specifications.
#
########################################################################

RELEASE="7.0"

USAGE="Usage: $0 [ -hV ] [--help] [--version] interface"
VERSTR="LFS ifdown, version ${RELEASE}"

while [ $# -gt 0 ]; do
   case "$1" in
      --help | -h)     help="y"; break ;;

      --version | -V)  echo "${VERSTR}"; exit 0 ;;

      -*)              echo "ifup: ${1}: invalid option" >&2
                       echo "${USAGE}" >& 2
                       exit 2 ;;

      *)               break ;;
   esac
done

if [ -n "$help" ]; then
   echo "${VERSTR}"
   echo "${USAGE}"
   echo
   cat << HERE_EOF
ifdown is used to bring down a network interface.  The interface
parameter, e.g. eth0 or eth0:2, must match the trailing part of the
interface specifications file, e.g. /etc/sysconfig/ifconfig.eth0:2.

HERE_EOF
   exit 0
fi
```

```
file=/etc/sysconfig/ifconfig.${1}

# Skip backup files
[ "${file}" = "${file%""~""}" ] || exit 0

. /lib/lsb/init-functions

if [ ! -r "${file}" ]; then
   log_warning_msg "${file} is missing or cannot be accessed."
   exit 1
fi

. ${file}

if [ "$IFACE" = "" ]; then
   log_failure_msg "${file} does not define an interface [IFACE]."
   exit 1
fi

# We only need to first service to bring down the interface
S=`echo ${SERVICE} | cut -f1 -d" "`

if ip link show ${IFACE} > /dev/null 2>&1; then
   if [ -n "${S}" -a -x "/lib/services/${S}" ]; then
     IFCONFIG=${file} /lib/services/${S} ${IFACE} down
   else
     MSG="Unable to process ${file}.  Either "
     MSG="${MSG}the SERVICE variable was not set "
     MSG="${MSG}or the specified service cannot be executed."
     log_failure_msg "$MSG"
     exit 1
  fi
else
   log_warning_msg "Interface ${1} doesn't exist."
fi

# Leave the interface up if there are additional interfaces in the device
link_status=`ip link show ${IFACE} 2>/dev/null`

if [ -n "${link_status}" ]; then
   if [ "$(echo "${link_status}" | grep UP)" != "" ]; then
      if [ "$(ip addr show ${IFACE} | grep 'inet ')" == ""  ]; then
         log_info_msg "Bringing down the ${IFACE} interface..."
         ip link set ${IFACE} down
         evaluate_retval
      fi
   fi
fi

# End /sbin/ifdown
```

Г.27. /lib/services/ipv4-static

```
#!/bin/sh
```

```
########################################################################
# Begin /lib/services/ipv4-static
#
# Description : IPV4 Static Boot Script
#
# Authors     : Nathan Coulson - nathan@linuxfromscratch.org
#               Kevin P. Fleming - kpfleming@linuxfromscratch.org
# Update      : Bruce Dubbs - bdubbs@linuxfromscratch.org
#
# Version     : LFS 7.0
#
########################################################################

. /lib/lsb/init-functions
. ${IFCONFIG}

if [ -z "${IP}" ]; then
   log_failure_msg "\nIP variable missing from ${IFCONFIG}, cannot
continue."
   exit 1
fi

if [ -z "${PREFIX}" -a -z "${PEER}" ]; then
   log_warning_msg "\nPREFIX variable missing from ${IFCONFIG}, assuming
24."
   PREFIX=24
   args="${args} ${IP}/${PREFIX}"

elif [ -n "${PREFIX}" -a -n "${PEER}" ]; then
   log_failure_msg "\nPREFIX and PEER both specified in ${IFCONFIG}, cannot
continue."
   exit 1

elif [ -n "${PREFIX}" ]; then
   args="${args} ${IP}/${PREFIX}"

elif [ -n "${PEER}" ]; then
   args="${args} ${IP} peer ${PEER}"
fi

if [ -n "${BROADCAST}" ]; then
   args="${args} broadcast ${BROADCAST}"
fi

case "${2}" in
   up)
      if [ "$(ip addr show ${1} 2>/dev/null | grep ${IP}/)" == "" ]; then

         # Cosmetic output not needed for multiple services
         if ! $(echo ${SERVICE} | grep -q " "); then
           log_info_msg2 "\n" # Terminate the previous message
         fi

         log_info_msg "Adding IPv4 address ${IP} to the ${1} interface..."
         ip addr add ${args} dev ${1}
```

```
         evaluate_retval
      else
         log_warning_msg "Cannot add IPv4 address ${IP} to ${1}.  Already
present."
      fi
   ;;

   down)
      if [ "$(ip addr show ${1} 2>/dev/null | grep ${IP}/)" != "" ]; then
         log_info_msg "Removing IPv4 address ${IP} from the ${1}
interface..."
         ip addr del ${args} dev ${1}
         evaluate_retval
      fi

      if [ -n "${GATEWAY}" ]; then
         # Only remove the gateway if there are no remaining ipv4 addresses
         if [ "$(ip addr show ${1} 2>/dev/null | grep 'inet ')" != "" ];
then
            log_info_msg "Removing default gateway..."
            ip route del default
            evaluate_retval
         fi
      fi
   ;;

   *)
      echo "Usage: ${0} [interface] {up|down}"
      exit 1
   ;;
esac

# End /lib/services/ipv4-static
```

Г.28. /lib/services/ipv4-static-route

```
#!/bin/sh
########################################################################
# Begin /lib/services/ipv4-static-route
#
# Description : IPV4 Static Route Script
#
# Authors     : Kevin P. Fleming - kpfleming@linuxfromscratch.org
#               DJ Lucas - dj@linuxfromscratch.org
# Update      : Bruce Dubbs - bdubbs@linuxfromscratch.org
#
# Version     : LFS 7.0
#
########################################################################

. /lib/lsb/init-functions
. ${IFCONFIG}

case "${TYPE}" in
   ("" | "network")
```

```
      need_ip=1
      need_gateway=1
   ;;

   ("default")
      need_gateway=1
      args="${args} default"
      desc="default"
   ;;

   ("host")
      need_ip=1
   ;;

   ("unreachable")
      need_ip=1
      args="${args} unreachable"
      desc="unreachable "
   ;;

   (*)
      log_failure_msg "Unknown route type (${TYPE}) in ${IFCONFIG}, cannot
continue."
      exit 1
   ;;
esac

if [ -n "${need_ip}" ]; then
   if [ -z "${IP}" ]; then
      log_failure_msg "IP variable missing from ${IFCONFIG}, cannot
continue."
      exit 1
   fi

   if [ -z "${PREFIX}" ]; then
      log_failure_msg "PREFIX variable missing from ${IFCONFIG}, cannot
continue."
      exit 1
   fi

   args="${args} ${IP}/${PREFIX}"
   desc="${desc}${IP}/${PREFIX}"
fi

if [ -n "${need_gateway}" ]; then
   if [ -z "${GATEWAY}" ]; then
      log_failure_msg "GATEWAY variable missing from ${IFCONFIG}, cannot
continue."
      exit 1
   fi
   args="${args} via ${GATEWAY}"
fi

if [ -n "${SOURCE}" ]; then
        args="${args} src ${SOURCE}"
```

```
fi

case "${2}" in
   up)
      log_info_msg "Adding '${desc}' route to the ${1} interface..."
      ip route add ${args} dev ${1}
      evaluate_retval
   ;;

   down)
      log_info_msg "Removing '${desc}' route from the ${1} interface..."
      ip route del ${args} dev ${1}
      evaluate_retval
   ;;

   *)
      echo "Usage: ${0} [interface] {up|down}"
      exit 1
   ;;
esac

# End /lib/services/ipv4-static-route
```

Додаток Д. Конфігураційні правила Udev

Правила з udev-lfs-188-3.tar.bz2 у цьому додатку перераховані для зручності. Встановлення типово виконується у Секції 6.61, “Udev-188(Витягнуті з systemd-188)”

Д.1. 55-lfs.rules

```
      # /etc/udev/rules.d/55-lfs.rules: Rule definitions for LFS.

# Core kernel devices

# This causes the system clock to be set as soon as /dev/rtc becomes
available.
SUBSYSTEM=="rtc", ACTION=="add", MODE="0644",
RUN+="/etc/rc.d/init.d/setclock start"
KERNEL=="rtc", ACTION=="add", MODE="0644", RUN+="/etc/rc.d/init.d/setclock
start"

# Comms devices

KERNEL=="ippp[0-9]*",       GROUP="dialout"
KERNEL=="isdn[0-9]*",       GROUP="dialout"
KERNEL=="isdnctrl[0-9]*",   GROUP="dialout"
KERNEL=="dcbri[0-9]*",      GROUP="dialout"
```

Додаток E. Ліцензії LFS

Ця книга підпадає під ліцензію Creative Commons Attribution-NonCommercial-ShareAlike 2.0 License.

Інструкції комп'ютера можуть бути витягнуті з книги під ліцензією MIT.

E.1. Ліцензія 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 ommons 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/*.

E.2. Ліцензія The MIT License

Copyright © 1999-2012 Gerard Beekmans

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions: The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Переклад — Сидор Ю.А. (crazy_red_goth@e-mail.ua)

29.11.2012 р.