Линукс с нуля

Версия 12.2-multilib

Дата публикации 1 сентября 2024

Создатель: Gerard Beekmans
Главный редактор: Bruce Dubbs
Редактор: Thomas Trepl
Автор перевода: Владимир Перцев

Линукс с нуля: Версия 12.2-multilib: Дата публикации 1 сентября 2024

Создатель: Gerard Beekmans, Главный редактор: Bruce Dubbs, Редактор: Thomas Trepl, Автор перевода: Владимир Перцев

Содержание

Предисловие

Предисловие

Мой путь к изучению и лучшему пониманию Linux начался в 1998 году. Я только что установил свой первый дистрибутив Linux и быстро увлекся его концепцией и философией.

Известно, что у задачи может быть несколько вариантов решения. То же самое можно сказать и о дистрибутивах Linux. Многие из них существовали годами. Некоторые всё еще существуют, некоторые превратились во что-то иное, а некоторые остались только в наших воспоминаниях. Все они выполняют задачи по-разному, чтобы удовлетворить потребности своей целевой аудитории. И я понял - раз существует так много всевозможных способов добиться поставленной цели, мне больше не нужно ограничивать себя какой-то одной реализацией. До появления Linux мы просто мирились с проблемами в других операционных системах, поскольку у нас не было выбора. Что есть, то есть, нравилось нам это или нет. С Linux появился выбор. Если вам что-то не понравилось, вы можете изменить это, к тому же, это всецело поощряется.

Я попробовал разные дистрибутивы, но так и не смог ни на одном остановиться. Они были отличными системами сами по себе. Это больше не было вопросом правильно или неправильно. Это стало делом личного вкуса. При всём разнообразии выбора не было ни одного дистрибутива, который был бы идеален для меня. Поэтому я решил создать свою собственную Linux-систему, которая бы полностью соответствовала моим личным предпочтениям.

Чтобы создать свою собственную систему, я решил скомпилировать всё из исходного кода вместо использования предварительно скомпилированных пакетов. Эта «идеальная» Linux-система должна была иметь сильные стороны других систем без их недостатков. Сначала эта мысль казалась пугающей. Но я придерживался идеи, что такая система должна быть создана.

Разобравшись с такими проблемами, как циклические зависимости и ошибки во время компиляции, я, наконец, создал собственную систему Linux. Она была полностью работоспособна и вполне пригодна для использования, как и любая другая Linux-система того времени. Но это было мое собственное творение. Было очень приятно собрать такую систему самому. Единственное, что было бы лучше, это создавать каждую часть программного обеспечения самостоятельно. Это было следующее, к чему я стремился

Когда я поделился своими идеями и опытом с другими членами сообщества Linux, стал очевиден явный интерес к ним. Вскоре стало понятно, что такие специально созданные Linux-системы служат не только для удовлетворения специфических потребностей пользователей, но и являются идеальной возможностью для обучения программистов и системных администраторов, чтобы улучшить их (существующие) навыки работы с Linux. Так родился проект *Linux From Scratch*.

Книга Linux From Scratch является ядром этого проекта. В ней содержится информация и инструкции, необходимые для разработки и создания собственной системы. Хотя эта книга представляет шаблон, который позволит создать правильно работающую систему, вы можете изменить инструкции по своему усмотрению, что отчасти является важной составляющей этого проекта. Вы всё контролируете; мы просто протягиваем руку помощи, чтобы вы начали свой собственный путь.

Я искренне надеюсь, что вы прекрасно проведете время, работая над своей собственной системой Linux From Scratch, и оцените ее многочисленные преимущества.

--
Gerard Beekmans
gerard@linuxfromscratch.org

Аудитория, на которую рассчитана эта книга

Есть множество причин, почему вам стоило бы прочитать эту книгу. Один из вопросов, который задают пользователи, звучит так: «Зачем тратить время на сборку Linux-системы вручную с нуля, если можно просто загрузить и установить существующую?»

Важной целью существования этого проекта является помощь в изучении того, как работает система Linux изнутри. Создание системы LFS помогает продемонстрировать, что заставляет работать Linux, как все работает вместе и зависит друг от друга. Одна из лучших вещей, которую может дать этот учебный опыт, — это возможность настроить систему Linux в соответствии с вашими уникальными потребностями.

Другое ключевое преимущество - LFS предоставляет более глубокий контроль, не полагаясь на чью-либо реализацию Linux. С LFS вы находитесь в кресле водителя, и *Вы* управляете каждым аспектом системы.

LFS позволяет создавать очень компактные системы Linux. При установке обычных дистрибутивов вам часто приходится устанавливать очень много программ, которые, вероятно, никогда не используются. Эти программы тратят ресурсы впустую. Вы можете возразить, что с сегодняшними жесткими дисками и процессорами такие ресурсы не имеют значения. Иногда, однако, вы все еще ограничены размером. Подумайте о загрузочных компакт-дисках, USB-накопителях и встраиваемых системах. Это области, в которых LFS может быть полезным.

Ещё одним преимуществом собственной сборки Linux является безопасность. При компиляции каждого компонента системы из исходного кода вы можете всё проверить и применить необходимые патчи. Больше не нужно ждать, когда кто-то другой скомпилирует пакет с требуемыми исправлениями. Если вы не изучите патч и не примените его самостоятельно, нет гарантий, что новый пакет будет собран корректно и устранит проблему.

Цель Linux From Scratch — создать законченную и пригодную для использования систему базового уровня. Если вы не хотите создавать свою собственную систему Linux с нуля, вы, тем не менее, можете извлечь пользу из информации, содержащейся в этой книге.

Есть много других веских причин для создания собственной системы LFS. В конце концов, образование, безусловно, является самой важной из них. Продолжая работать с LFS, вы откроете для себя силу, которую действительно приносят информация и знания.

Целевые архитектуры LFS

Основными целевыми архитектурами LFS являются процессоры AMD/Intel x86 (32-разрядные) и x86_64 (64-разрядные). Однако, известно, что инструкции, приведенные в этой книге, с некоторыми изменениями работают с процессорами Power PC и ARM. Для создания системы, использующей один из этих процессоров, основным предварительным условием, в дополнение к описанным на следующей странице, является существующая система Linux, например, собранная ранее LFS, Ubuntu, Red Hat/Fedora, SuSE или другой дистрибутив, ориентированный на имеющуюся у вас архитектуру. Также обратите внимание, что 32-разрядный дистрибутив можно установить и использовать в качестве хост-системы на 64-разрядном компьютере AMD/Intel.

При сборке LFS выигрыш от сборки на 64-битной системе по сравнению с 32-битной системой минимален. Например, в тестовой сборке LFS-9.1 в системе на базе процессора Core i7-4790 с использованием 4 ядер были получены следующие статистические данные:

```
Архитектура     Время сборки    Размер сборки
x86             239.9 минут     3.6 ГБ
x64             233.2 минут     4.4 ГБ
```

Как видите, на том же оборудовании 64-битная сборка всего на 3% быстрее и на 22% больше, чем 32-битная. Если вы планируете использовать LFS в качестве LAMP-сервера или брандмауэра, 32-разрядной сборки может быть достаточно. С другой стороны, для сборки и/или запуска некоторых пакетов в BLFS теперь требуется более 4 ГБ ОЗУ, поэтому, если вы планируете использовать LFS в качестве настольной ОС, авторы LFS рекомендуют собирать 64-битную систему.

По умолчанию 64-разрядная сборка LFS, считается «чистой» 64-разрядной системой. То есть она поддерживает только 64-разрядные исполняемые файлы. Сборка «multilib» системы требует компиляции многих программ дважды - один раз для 32-битной и один раз для 64-битной. Напрямую в книге данная опция не поддерживается, потому что это будет только мешать образовательной цели этой книги, предлагающей инструкции, необходимые для сборки базовой системы. Некоторые редакторы LFS/BLFS поддерживают ответвление LFS для multilib, которое доступно по адресу *https://book.linuxfromscratch.ru/12.2-ml/systemv*, но это более продвинутая тема.

Требования к уровню подготовки

Создание системы LFS — непростая задача. Это требует определенного уровня знаний системного администрирования Unix для решения проблем и правильного выполнения перечисленных команд. В частности, как абсолютный минимум, Вы должны уметь пользоваться командной оболочкой для копирования или перемещения файлов и каталогов, просмотра содержимого каталогов и файлов и изменения текущего каталога. Также ожидается, что у вас есть достаточные знания об использовании и установке программного обеспечения в Linux.

Поскольку книга LFS предполагает наличие *хотя бы этого* базового уровня навыков, различные форумы поддержки LFS вряд ли смогут предоставить вам большую помощь в этих вопросах. Вы обнаружите, что ваши вопросы, касающиеся таких базовых знаний, скорее всего, останутся без ответа (или вас просто направят к списку основных материалов для предварительного ознакомления).

Перед созданием системы LFS мы настоятельно рекомендуем прочитать следующие статьи:

- HOWTO по сборке программного обеспечения *https://tldp.org/HOWTO/Software-Building-HOWTO.html*

  Это исчерпывающее руководство по сборке и установке «универсальных» программ Unix под Linux. Несмотря на то что руководство написано достаточно давно, оно по-прежнему дает хороший обзор основных методов, применяемых для сборки и установки программного обеспечения.
- Руководство для начинающих по установке из исходников *https://moi.vonos.net/linux/beginners-installing-from-source/*

  В этом руководстве содержится хороший обзор основных навыков и методов, необходимых для сборки программ из исходного кода

О выпуске Multilib

Что такое Multilib

Сегодня большинство систем в мире имеют размер слова 64 бита. Размер слова - это количество бит, которое может быть использовано одновременно наиболее эффективным способом. Предыдущие архитектуры процессоров x86 имели размер слова 32 бита, это означает, что у них другое понимание того, как лучше выравнивать данные в памяти, а также у них свой механизм обращения к памяти другого размера. Хотя набор команд 64-разрядных процессоров довольно схож с инструкциями 32-разрядных процессоров, двоичные файлы (объекты, скомпилированные в машинный код) не могут быть непосредственно выполнены в 64-разрядных системах.

Multilib - это механизм, обеспечивающий поддержку 32-разрядных двоичных файлов, чтобы они могли выполняться на современных 64-разрядных процессорах.

Зачем делать Multilib-систему?

С образовательной точки зрения LFS в своей классической форме, вероятно, является лучшим способом увидеть, как система Linux собирается из исходного кода. Нет необходимости смешивать различные архитектуры. Но когда необходимо запустить 32-битные двоичные файлы и вы не хотите собирать полностью 32-битную систему (что в наши дни было бы пустой тратой ресурсов), тогда LFS-multilib — это вариант. Примерами такой необходимости могут быть

- Программное обеспечение с закрытым исходным кодом/без исходного кода доступное только для 32-разрядной версии. Это может относиться к драйверу принтера или любым другим драйвером оборудования, двоичные файлы которого предоставляет компания-разработчик. Если у вас есть такой драйвер, LFS-multilib может помочь вам запустить его
- Если вы хотите выйти далеко за рамки LFS и настроить платформу виртуализации, такую как VirtualBox, вам понадобится поддержка multilib
- или даже просто потому, что вы можете

Multilib версия LFS дополняет то, что было сказано в предыдущем разделе, когда речь шла о целевых архитектурах. С одной стороны, multilib версия ориентирована и *ограничена* только архитектурой x86_64, с другой стороны, она *расширяет* инструкции для использования обеих возможных архитектур, а именно 32-разрядной как и 64-разрядной.

Сборка multilib-системы также немного выходит за рамки базового образовательного подхода LFS, который заключается в том, чтобы показать вам, как собрать систему Linux с нуля. Для этого не требуется поддержка дополнительных архитектур, отличных от архитектуры вашей системы по умолчанию. Если ранее вы не собирали систему по классической книге LFS, вам рекомендуется сделать это перед использованием этого издания.

Сборка Multilib системы

Сборка multilib системы не сильно отличается от сборки системы с использованием классической книги LFS. Помимо некоторых настроек, наиболее заметным отличием является то, что multilib требует компиляции некоторых приложений до трех раз: один раз для основной 64-разрядной архитектуры, второй - для 32-разрядной архитектуры (m32) и еще раз для 32-разрядной архитектуры с 32-битным доступом к памяти и 64-битным набором команд (mx32).

Продолжайте только в том случае, если ваша система удовлетворяете следующим требованиям:

- у вас компьютер, совместимый с x86_64

Примечание

Если у вас есть доступ к конфигурации ядра вашей системы, вам потребуется установить параметры: *CONFIG_IA32_EMULATION=y* , чтобы выполнить сборку для *m32* и *CONFIG_X86_X32=y* , чтобы собрать пакет для *mx32*.

Если у вас нет доступа к конфигурации ядра вашей системы, вы можете проверить совместимость, выполнив следующие команды:

```
echo 'int main(){}' > dummy.c
gcc -m32 dummy.c
./a.out

gcc -mx32 dummy.c
./a.out
```

Если какая-либо из команд **./a.out** возвращает сообщения "Exec format error", то ваше ядро собрано без поддержки нужных параметров для этой архитектуры в рамках LFS, поэтому вам, вероятно, следует собрать систему LFS без Multilib, но с необходимыми возможностями ядра и использовать ее для сборки системы Multilib.

- у вас уже есть некоторый опыт работы с LFS
- вам необходима поддержка 32-битных версий библиотек

Если выполнены все три условия, продолжайте собирать LFS в режиме multilib.

LFS и стандарты

Структура LFS максимально соответствует стандартам Linux. Первичными стандартами являются:

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

  LSB имеет четыре отдельных стандарта: Core, Desktop, Runtime Languages и Imaging. Некоторые части спецификаций Core и Desktop зависят от архитектуры. Есть также две области не являющиеся обязательными: Gtk3 и Graphics. LFS старается соответствовать стандартам LSB, для архитектур IA32 (32-bit x86) или AMD64 (x86_64), рассмотренных в предыдущем разделе.

Примечание

Многие не согласны с требованиями LSB. Основные цели стандартов - быть уверенным в том, что проприетарное ПО будет правильно установлено и сможет корректно работать на совместимой системе. Поскольку в LFS установка программ выполняется с помощью сборки из исходного кода, у пользователя имеется полный контроль над тем, какие пакеты ему необходимы, вы можете не устанавливать некоторые пакеты, определяемые в LSB.

Создать законченную систему, которая пройдет сертификационные тесты LSB «с нуля», возможно, но этого нельзя сделать без установки множества дополнительных пакетов, которые выходят за рамки этой книги. Однако, инструкции по их установке можно найти в книге BLFS.

Пакеты LFS, которые необходимы для удовлетворения требований LSB

LSB Core:	Bash, Bc, Binutils, Coreutils, Diffutils, File, Findutils, Gawk, Grep, Gzip, M4, Man-DB, Ncurses, Procps, Psmisc, Sed, Shadow, Tar, Util-linux, Zlib

LSB Desktop:	Нет
LSB Runtime Languages:	Perl, Python
LSB Imaging:	Нет
LSB Gtk3 и LSB Graphics (Необязательные):	Нет

Пакеты, поставляемые BLFS, необходимые для удовлетворения требований LSB

LSB Core:	At, Batch (часть At), Cpio, Ed, Fcrontab, LSB-Tools, NSPR, NSS, PAM, Pax, Sendmail (или Postfix, или Exim), time
LSB Desktop:	Alsa, ATK, Cairo, Desktop-file-utils, Freetype, Fontconfig, Gdk-pixbuf, Glib2, GTK+2, Icon-naming-utils, Libjpeg-turbo, Libpng, Libtiff, Libxml2, MesaLib, Pango, Xdg-utils, Xorg
LSB Runtime Languages:	Libxml2, Libxslt
LSB Imaging:	CUPS, Cups-filters, Ghostscript, SANE
LSB Gtk3 и LSB Graphics (Необязательные):	GTK3+

Пакеты, не поставляемые LFS или BLFS, необходимые для удовлетворения требований LSB

LSB Core:	Нет
LSB Desktop:	Qt4 (но предоставляется Qt5)
LSB Runtime Languages:	Нет
LSB Imaging:	Нет
LSB Gtk3 и LSB Graphics (Необязательные):	Нет

Информация о пакетах, используемых в этой книге

Целью LFS является создание законченной и пригодной для использования базовой системы, которая содержит все пакеты, необходимые для её функционирования, состоящую при этом из относительно небольшого набора программ, и возможности которой можно расширять в зависимости от потребностей пользователя. Это не означает, что LFS является самой маленькой из возможных систем. В систему включено несколько важных пакетов, которые не являются обязательными. Приведенный ниже список объясняет почему в книгу включен тот или иной пакет.

- Acl

  Access Control List или ACL — список управления доступом, который определяет, кто или что может получать доступ к объекту (программе, процессу или файлу), и какие именно операции разрешено или запрещено выполнять субъекту (пользователю, группе пользователей).Данный пакет содержит утилиты для администрирования списков управления доступом, которые используются для определения дискреционных прав доступа к файлам и каталогам.

- Attr

  Этот пакет содержит программы для управления расширенными атрибутами объектов файловой системы.

- Autoconf

Этот пакет содержит программы для создания сценариев оболочки, которые могут выполнять автоматическую настройку исходного кода из шаблона разработчика. Он часто необходим для повторной компиляции пакета после обновления процедур сборки.

- Automake

  Этот пакет содержит программы для создания Make-файлов из шаблона. Он также необходим для повторной компиляции пакета после обновления процедур сборки.

- Bash

  Этот пакет удовлетворяет требования LSB по предоставлению интерфейса Bourne Shell для системы. Он был выбран среди других пакетов оболочки из-за его повсеместного использования и широких возможностей.

- Bc

  Этот пакет предоставляет язык числовой обработки произвольной точности. Он необходимым для сборки ядра Linux

- Binutils

  Этот пакет содержит компоновщик, ассемблер и другие инструменты для работы с объектными файлами. Программы в этом пакете необходимы для компиляции большинства пакетов в системе LFS.

- Bison

  Этот пакет содержит GNU-версию yacc (Yet Another Compiler Compiler), необходимого для сборки некоторых пакетов в LFS.

- Bzip2

  Этот пакет содержит программы для сжатия и распаковки файлов. Используется для распаковки множества пакетов LFS.

- Check

  Этот пакет содержит тестовую обвязку для других программ.

- Coreutils

  Этот пакет содержит ряд программ для просмотра файлов и каталогов, и управления ими. Эти программы необходимы для управления файлами через командную строку и для сборки каждого пакета в LFS.

- DejaGNU

  Этот пакет предоставляет фреймворк для тестирования других программ.

- Diffutils

  Этот пакет содержит программы, которые показывают различия между файлами или каталогами. Их можно использовать для создания патчей, а также они применяются во многих процедурах сборки

- E2fsprogs

  Этот пакет содержит утилиты для работы с файловыми системами ext2, ext3 и ext4. Это наиболее распространенные и тщательно протестированные файловые системы, поддерживаемые Linux

- Expat

  Этот пакет содержит небольшую библиотеку разбора XML.Она необходима для модуля Perl XML::Parser.

- Expect

  Этот пакет содержит инструменты для автоматизации и тестирования, и является расширением к скриптовому языку Tcl, для многих интерактивных приложений. Он обычно используется для тестирования других пакетов.

- File

  Этот пакет содержит утилиту для определения типа файла или файлов. Некоторым пакетам она нужна в сценариях сборки.

- Findutils

  Этот пакет предоставляет программы для поиска файлов. Он используется во многих сценариях сборки пакетов.

- Flex

  Этот пакет содержит утилиту для генерации программ, распознающих шаблоны в тексте. Это версия GNU программы lex (лексический анализатор). Пакет необходим для сборки некоторых пакетов LFS.

- Gawk

  Этот пакет содержит программы для работы с текстовыми файлами. Это GNU версия awk (Aho-Weinberg-Kernighan). Он используется во многих сценариях сборки пакетов.

- GCC

  Это коллекция компиляторов Gnu. Он содержит компиляторы C и C++, а также несколько других компиляторов, поддержка которых не предусмотрена в LFS.

- GDBM

  Этот пакет содержит библиотеку GNU Database Manager. Он используется пакетом Man-DB

- Gettext

  Этот пакет содержит утилиты и библиотеки для интернационализации и локализации многочисленных пакетов.

- Glibc

  Этот пакет содержит основную библиотеку C. Программы Linux не будут работать без неё.

- GMP

  Этот пакет содержит математические библиотеки, предоставляющие полезные функции для вычислений с плавающей точкой. Требуется для сборки GCC.

- Gperf

  Этот пакет содержит программу, которая генерирует идеальную хеш-функцию из набора ключей. Необходим для пакета Udev .

- Grep

  Этот пакет содержит программы для поиска по файлам. Пакет используется в скриптах сборки большинства пакетов.

- Groff

  Этот пакет содержит программы для обработки и форматирования текста. Одной из важнейших функций этих программ является форматирование man страниц.

- GRUB

Это загрузчик операционной системы (GRand Unified Bootloader). Самый гибкий из нескольких доступных загрузчиков.

- Gzip

  Этот пакет содержит программы для сжатия и распаковки файлов. Он необходим для распаковки множества пакетов в LFS.

- Iana-etc

  Этот пакет предоставляет данные для сетевых служб и протоколов. Он необходим для обеспечения правильных сетевых возможностей.

- Inetutils

  Этот пакет содержит программы для базового сетевого администрирования.

- Intltool

  Этот пакет содержит инструменты для извлечения переводимых строк из исходных файлов.

- IProute2

  Этот пакет содержит программы для базовой и расширенной работы в сетях IPv4 и IPv6. Он был выбран среди других распространенных пакетов сетевых инструментов (net-tools) из-за его поддержки IPv6.

- Kbd

  Этот пакет содержит таблицы раскладок, утилиты управления клавиатурой для неамериканских клавиатур, кроме этого, с ним поставляется большой набор консольных шрифтов.

- Kmod

  Этот пакет содержит программы, необходимые для администрирования модулей ядра Linux.

- Less

  Этот пакет содержит очень хороший просмотрщик текстовых файлов, который позволяет использовать прокрутку верх/вниз при просмотре. Многие пакеты используют его для постраничного вывода.

- Libcap

  Этот пакет реализует интерфейсы пользовательского пространства для возможностей POSIX 1003.1e, доступных в ядре Linux.

- Libelf

  Проект elfutils предоставляет библиотеки и инструменты для файлов ELF и данных DWARF. Большинство утилит в этом пакете доступны в других пакетах, но эта библиотека необходима для сборки ядра Linux с использованием стандартной (и наиболее эффективной) конфигурации.

- Libffi

  Этот пакет реализует переносимый программный интерфейс высокого уровня для различных соглашений о вызовах. Некоторые программы могут не знать во время компиляции, какие аргументы должны быть переданы в функцию. Например, интерпретатору во время выполнения может быть сообщено о количестве и типах аргументов, используемых для вызова данной функции. Libffi можно использовать как мост от интерпретатора к скомпилированному коду.

- Libpipeline

  Пакет Libpipeline содержит библиотеку для гибкого и удобного управления конвейерами подпроцессов. Она необходима для Man-DB.

- Libtool

Этот пакет содержит сценарий поддержки универсальной библиотеки GNU. Он объединяет сложность использования общих библиотек в согласованный переносимый интерфейс. Библиотека необходима наборам тестов в других пакетах LFS.

- Libxcrypt

  Этот пакет предоставляет библиотеку `libcrypt`, необходимую различным пакетам (в частности, Shadow) для хеширования паролей. Он заменяет устаревшую реализацию `libcrypt` в Glibc.

- Linux Kernel

  Этот пакет является ядром операционной системой.

- M4

  Этот пакет содержит текстовый макропроцессор, полезный в качестве инструмента сборки для других программ.

- Make

  Этот пакет содержит программу для управления сборкой пакетов. При сборке она необходима почти для каждого пакета в LFS.

- Man-DB

  Этот пакет содержит программы для поиска и просмотра справочных страниц. Он был выбран вместо пакета man из-за превосходных возможностей интернационализации. Содержит man.

- Man-pages

  Этот пакет содержит основные справочные(man) страницы Linux.

- Meson

  Этот пакет предоставляет программный инструмент для автоматизации создания программного обеспечения. Основная цель Meson — свести к минимуму количество времени, которое разработчики программного обеспечения должны тратить на настройку своей системы сборки. Требуется для сборки Systemd, а также многих пакетов BLFS.

- MPC

  Этот пакет содержит функции для арифметики комплексных чисел. Необходим GCC.

- MPFR

  Этот пакет содержит функции для арифметики произвольной точности. Необходим GCC.

- Ninja

  Этот пакет предоставляет небольшую систему сборки, ориентированную на скорость. Он предназначен для того, чтобы его входные файлы генерировались системой сборки более высокого уровня, и для максимально быстрого запуска сборок. Необходим для Meson.

- Ncurses

  Этот пакет содержит библиотеки для независимой от терминала обработки символьных экранов. Он часто используется для управления курсором в меню. Необходим ряду пакетов в LFS.

- Openssl

  Этот пакет содержит инструменты управления и библиотеки, относящиеся к криптографии. Они предоставляют криптографические функций другим пакетам, включая ядро Linux.

- Patch

Этот пакет содержит программу для изменения или создания файлов путем применения файла *patch*, обычно создаваемого программой diff. Он необходим процедуре сборки для некоторых пакетов LFS.

- Perl

  Этот пакет является интерпретатором языка PERL. Он необходим для установки и тестирования некоторых пакетов LFS.

- Pkgconf

  Этот пакет содержит программу, которая помогает настраивать флаги компилятора и компоновщика для библиотек разработки. Программа может быть использована в качестве замены **pkg-config**, который необходим системе сборки многих пакетов. Он поддерживается более активно и развивается немного быстрее, чем оригинальный пакет Pkg-config.

- Procps-NG

  Этот пакет содержит программы для мониторинга процессов. Набор полезен для системного администрирования, а также используются загрузочными сценариями LFS.

- Psmisc

  Этот пакет содержит программы для отображения информации о запущенных процессах. Этот набор программ полезен для системного администрирования.

- Python 3

  Этот пакет предоставляет интерпретируемый язык программирования, философия которого делает упор на удобочитаемость кода.

- Readline

  Этот пакет представляет собой набор библиотек, предлагающих возможности редактирования командной строки и средства для работы с историей команд. Используется командным интерпретатором Bash.

- Sed

  Этот пакет позволяет редактировать текст, не открывая его в текстовом редакторе. Он необходим сценариям настройки многих пакетов LFS.

- Shadow

  Этот пакет содержит программы для безопасной обработки паролей.

- Sysklogd

  Этот пакет содержит программы для регистрации системных сообщений, которые генерируются ядром или процессами демона при возникновении необычных событий.

- SysVinit

  Этот пакет содержит систему инициализации init, родительской элемент всех остальных процессов в работающей системе Linux.

- Udev

  Этот пакет представляет собой диспетчер устройств. Он контролирует разрешения, имена и символические ссылки устройств в каталоге /dev, когда устройства добавляются или удаляются из системы.

- Tar

  Этот пакет предоставляет возможность архивирования и извлечения практически всех пакетов, используемых в LFS.

- Tcl

  Этот пакет содержит командный язык инструментов, используется во многих наборах тестов.

- Texinfo

  Этот пакет предоставляет программы для чтения, записи и преобразования информационных страниц. Используется в процедурах установки многих пакетов LFS.

- Util-linux

  Этот пакет содержит различные служебные программы. Среди них утилиты для работы с файловыми системами, консолями, разделами и сообщениями.

- Vim

  Этот пакет содержит редактор. Его выбрали из-за совместимости с классическим редактором vi и огромного количества возможностей. Редактор является очень личным выбором для каждого пользователя. По желанию можно заменить любым другим редактором.

- Wheel

  Этот пакет содержит модуль Python, который представляет собой эталонную реализацию механизма упаковки Python.

- XML::Parser

  Этот пакет представляет собой модуль Perl, который взаимодействует с Expat.

- XZ Utils

  Этот пакет содержит программы для сжатия и распаковки файлов. Он обеспечивает высокое сжатие и используется для распаковки пакетов в формате XZ или LZMA.

- Zlib

  Этот пакет содержит процедуры сжатия и распаковки, используемые некоторыми программами.

- Zstd

  Этот пакет содержит процедуры сжатия и распаковки, используемые некоторыми программами. Он обеспечивает высокие коэффициенты сжатия и очень широкий диапазон компромиссов между сжатием и скоростью.

Оформление

Чтобы облегчить понимание, в этой книге используются условные обозначения. Этот раздел содержит примеры оформления, используемые в Linux From Scratch.

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

Такое оформление предназначено для ввода именно так, как показано, если иное не сказано в тексте рядом. Это оформление также используется в разделах пояснений, чтобы указать, на какую команду ссылается.

В некоторых случаях логическая строка расширяется до двух или более физических строк с обратной косой чертой в конце строки.

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

Обратите внимание, что за обратной косой чертой должен следовать перевод строки. Другие символы, такие как пробелы или символы табуляции, приведут к неправильным результатам.

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

Такое оформление (текст фиксированной ширины) показывает вывод на экран, как правило, в результате выполнения команд. Этот формат также используется для отображения имен файлов, таких как `/etc/ld.so.conf` .

Примечание

Пожалуйста, настройте свой браузер для отображения текста фиксированной ширины с хорошим моноширинным шрифтом, с помощью которого вы сможете четко различать символы `Il1` или `00`.

Акцент

Эта форма текста используется в книге для нескольких целей. Его основная цель — подчеркнуть важные моменты.

https://mirror.linuxfromscratch.ru/

Этот формат используется для гиперссылок как на сайт сообщества LFS, так и на внешние ресурсы. Может включать справочную информацию, места загрузки и веб-сайты.

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

Этот формат используется при создании файлов конфигурации. Первая команда указывает системе создать файл `$LFS/etc/group` из всего, что введено далее, пока не встретится последовательность End Of File (EOF). Поэтому весь этот раздел обычно печатается как есть.

<ЗАМЕНЯЕМЫЙ ТЕКСТ>

Этот формат используется для текста, который не должен быть напечатан так, как отображается, или для операций копирования и вставки.

[НЕОБЯЗАТЕЛЬНЫЙ ТЕКСТ]

Этот формат используется для текста, который является необязательным.

passwd(5)

Этот формат используется для ссылки на определенную страницу руководства (man). Число в скобках указывает на конкретный раздел внутри руководств. Например, у **passwd** есть две справочные страницы. В соответствии с инструкциями по установке LFS эти две справочные страницы будут расположены в `/usr/share/man/man1/passwd.1` и `/usr/share/man/man5/passwd.5` . Когда в книге используется *passwd(5)*, имеется в виду конкретно `/usr/share/man/man5/passwd.5` . **man passwd** напечатает первую найденную справочную страницу, совпадающую с «passwd» - это будет `/usr/share/man/man1/passwd.1` . В этом примере вам нужно будет запустить **man 5 passwd**, чтобы прочитать указанную страницу. Обратите внимание, что большинство справочных страниц не имеют повторяющихся страниц в разных разделах. Поэтому обычно достаточно **man** ***<имя программы>***. В книге LFS ссылки на справочные страницы также являются гиперссылками, поэтому нажатие на такую ссылку откроет справочную страницу, в формате HTML, со *страниц руководства Arch Linux*.

Структура

Эта книга разделена на несколько частей.

Часть I - Введение

Эта часть содержит важные замечания о том, как выполнить установку LFS. Также здесь представлена метаинформация о книге

Часть II - Подготовка к сборке

Часть II описывает, как подготовиться к процессу сборки — создать разделы, загрузить пакеты и выполнить компиляцию временных инструментов.

Часть III - Создание кросс-тулчейна LFS и временных инструментов

Часть III содержит инструкции по созданию инструментов, необходимых для создания конечной системы LFS.

Часть IV - Сборка системы LFS

Часть IV проводит читателя через сборку системы LFS—компиляцию и установку всех пакетов один за другим, настройку сценариев загрузки и установку ядра. Полученная в результате система Linux является основой, на которой можно собрать другое программное обеспечение для расширения возможностей системы. В конце этой книги есть простой в использовании справочник со списком всех программ, библиотек и важных файлов, которые были установлены.

Часть V - Приложения

Часть V содержит информацию о самой книге, включая акронимы и термины, благодарности, зависимости пакетов, список загрузочных сценариев LFS, лицензии на распространение книги и исчерпывающий указатель пакетов, программ, библиотек и сценариев.

Ошибки и рекомендации по безопасности

Программное обеспечение, используемое для создания системы LFS, постоянно обновляется и совершенствуется. Предупреждения безопасности и исправления ошибок могут появиться после выхода книги LFS. Чтобы проверить, нуждаются ли пакеты или инструкции в этом выпуске LFS в каких-либо изменениях для устранения уязвимостей в системе безопасности или исправления других ошибок, посетите *https://mirror.linuxfromscratch.ru/lfs/errata/12.2/*, прежде чем приступить к сборке. Вы должны внести требуемые изменения и применить их к соответствующему разделу книги по мере сборки системы LFS.

Кроме того, редакторы Linux From Scratch ведут список уязвимостей безопасности, обнаруженных *после* выхода книги. Чтобы проверить наличие каких-либо известных уязвимостей безопасности, посетите *https://mirror.linuxfromscratch.ru/lfs/advisories/*, прежде чем продолжить сборку. И, если вы будете использовать систему LFS в качестве реальной настольной или серверной системы, вам следует обращаться к рекомендациям и устранять любые уязвимости в системе безопасности, даже когда система LFS полностью собрана.

Часть I. Введение

Глава 1. Введение

1.1. Как собрать систему LFS

Система LFS будет собрана с использованием уже установленного дистрибутива Linux (например, Debian, OpenMandriva, Fedora или openSUSE). Существующая система Linux (хост) будет использоваться в качестве отправной точки для предоставления необходимых программ, включая компилятор, компоновщик и оболочку, для создания новой системы. Выберите опцию «разработка» во время установки дистрибутива, чтобы получить доступ к этим инструментам.

Примечание

Существует множество способов установки дистрибутива Linux, и значения по умолчанию обычно не оптимальны для сборки системы LFS. Предложения по настройке дистрибутива смотрите: *https://mirror.linuxfromscratch.ru/hints/downloads/files/partitioning-for-lfs.txt*.

В качестве альтернативы установке отдельного дистрибутива на свой компьютер вы можете использовать LiveCD другого дистрибутива.

Глава 2 этой книги содержит информацию, о том, как создать новые разделы Linux и файловую систему, где будет скомпилирована и установлена новая система LFS. Глава 3 содержит информацию, о том, какие пакеты и исправления необходимо загрузить для сборки системы LFS и как их хранить на файловой системе. Глава 4 освещает вопросы настройки рабочего окружения. Пожалуйста, внимательно прочитайте Глава 4, так как в ней объясняется несколько важных моментов, о которых вам необходимо знать, прежде чем вы начнёте работать со следующими главами.

Глава 5 содержит информацию об установке первоначального набора инструментов (binutils, gcc и glibc) с использованием методов кросс-компиляции для изоляции новых инструментов от хост-системы.

Глава 6 рассказывает, как выполнить кросс-компиляцию базовых утилит с использованием только что собранного временного набора инструментов.

В Глава 7 будет осуществлен переход в среду **chroot**, где мы будет использовать новые инструменты для сборки остальных инструментов, необходимых для создания конечной системы.

Эта попытка изолировать новую систему от основного дистрибутива поначалу может показаться чрезмерной. Полное техническое обоснование того, почему это сделано именно так, приведено в разделе Технические примечания по сборочным инструментам.

В Глава 8 будет собрана полноценная система LFS. Еще одно преимущество среды chroot заключается в том, что она позволяет вам продолжать использовать хост-систему во время сборки LFS. Ожидая завершения компиляции пакетов, вы можете продолжать пользоваться своим компьютером в обычном режиме.

Чтобы завершить установку, в Глава 9 происходит настройка базовой конфигурации системы, в Глава 10 настраиваются ядро и загрузчик. Глава 11 содержит информацию о том как расширить возможности системы LFS. После выполнения шагов, описанных в этой главе, компьютер будет готов к загрузке в новую систему LFS.

Здесь описан процесс сборки системы в двух словах. Подробная информация о каждом шаге рассматривается в следующих главах и описаниях пакетов. Элементы, которые кажутся сложными сейчас, будут разъяснены позже, и все встанет на свои места, по мере прочтения книги.

1.2. Что нового с момента последнего релиза

Ниже приведен список пакетов, обновленных с момента предыдущего выпуска книги.

Обновлены:

- Automake-1.17
- Bash-5.2.32
- Bc-6.7.6
- Binutils-2.43.1
- Coreutils-9.5
- E2fsprogs-1.47.1
- Expat-2.6.2
- Findutils-4.10.0
- GCC-14.2.0
- GDBM-1.24
- Gettext-0.22.5
- Glibc-2.40
- Iana-Etc-20240806
- IPRoute2-6.10.0
- Jinja2-3.1.4
- Kmod-33
- Less-661
- Libcap-2.70
- Libelf из Elfutils-0.191
- Libffi-3.4.6
- Linux-6.10.5
- Lz4-1.10.0
- Man-DB-2.12.1
- Man-pages-6.9.1
- Meson-1.5.1
- Ncurses-6.5
- Ninja-1.12.1
- OpenSSL-3.3.1
- Perl-5.40.0
- Pkgconf-2.3.0
- Psmisc-23.7
- Python-3.12.5
- Readline-8.2.13
- Setuptools-72.2.0

- Shadow-4.16.0
- Sysklogd-2.6.1
- Systemd-256.4
- SysVinit-3.10
- Tcl-8.6.14
- Udev из Systemd-256.4
- Util-linux-2.40.2
- Vim-9.1.0660
- Wheel-0.44.0
- Xz-5.6.2
- Zstd-1.5.6

Добавлены:

- Lz4-1.10.0

Удалены:

- bash-5.2.21-upstream_fixes-1.patch
- readline-8.2-upstream_fixes-3.patch

1.3. Журнал изменений

Это версия 12.2-multilib книги Linux From Scratch от 1 сентября 2024. Если этой книге больше шести месяцев, возможно, уже доступна более новая, улучшенная версия. Чтобы узнать это, проверьте одно из зеркал *https://mirror.linuxfromscratch.ru/mirrors.html*.

Ниже приведен список изменений, внесенных с момента предыдущего выпуска книги.

Список изменений:

- 2024-08-23
  - [xry111] - Update to lfs-bootscripts-20240825. Only trivial non-functional changes.
- 2024-08-23
  - [xry111] - Update to lfs-bootscripts-20240823, to fix an issue causing VT 2-6 not affected by the FONT= setting in /etc/sysconfig/console.
- 2024-08-17
  - [bdubbs] - Update to setuptools-72.2.0. Fixes *#5542*.
  - [bdubbs] - Update to kmod-33. Fixes *#5540*.
  - [bdubbs] - Update to binutils-2.43.1. Fixes *#5543*.
  - [bdubbs] - Update to linux-6.10.5. Fixes *#5541*.
- 2024-08-15
  - [bdubbs] - Update to iana-etc-20240806. Addresses *#5006*.
  - [bdubbs] - Update to pkgconf-2.3.0. Fixes *#5537*.
  - [bdubbs] - Update to python3-3.12.5. Fixes *#5538*.

  - [bdubbs] - Update to linux-6.10.4. Fixes *#5539*.
- 2024-08-05
  - [bdubbs] - Update to bash-5.2.32. Fixes *#5532*.
  - [bdubbs] - Update to iana-etc-20240801. Addresses *#5006*.
  - [bdubbs] - Update to vim-9.1.0660. Addresses *#4500*.
  - [bdubbs] - Update to binutils-2.43. Fixes *#5535*.
  - [bdubbs] - Update to linux-6.10.3. Fixes *#5534*.
  - [bdubbs] - Update to readline-8.2.13. Fixes *#5533*.
  - [bdubbs] - Update to wheel-0.44.0. Fixes *#5536*.
- 2024-08-01
  - [bdubbs] - Update to gcc-14.2.0. Fixes *#5530*.
  - [bdubbs] - Update to iana-etc-20240723. Addresses *#5006*.
  - [bdubbs] - Update to glibc-2.40. Fixes *#5529*.
  - [bdubbs] - Update to iproute2-6.10.0. Fixes *#5523*.
  - [bdubbs] - Update to linux-6.10.2. Fixes *#5521*.
  - [bdubbs] - Update to lz4-1.10.0. Fixes *#5526*.
  - [bdubbs] - Update to meson-1.5.1. Fixes *#5527*.
  - [bdubbs] - Update to setuptools-72.1.0. Fixes *#5531*.
  - [bdubbs] - Update to sysklogd-2.6.1. Fixes *#5522*.
  - [bdubbs] - Update to sysvinit-3.10. Fixes *#5528*.
- 2024-07-15
  - [bdubbs] - Update to iana-etc-20240701. Addresses *#5006*.
  - [bdubbs] - Update to vim-9.1.0580. Addresses *#4500*.
  - [bdubbs] - Update to automake-1.17. Fixes *#5520*.
  - [bdubbs] - Update to gdbm-1.24. Fixes *#5515*.
  - [bdubbs] - Update to linux-6.9.9. Fixes *#5517*.
  - [bdubbs] - Update to less-661. Fixes *#5513*.
  - [bdubbs] - Update to meson-1.5.0. Fixes *#5519*.
  - [bdubbs] - Update to setuptools-70.3.0. Fixes *#5514*.
  - [bdubbs] - Update to util-linux-2.40.2. Fixes *#5516*.
- 2024-07-01
  - [bdubbs] - Update lfs-bootscripts to only output escape sequences to a terminal.
- 2024-07-01
  - [bdubbs] - Update to iana-etc-20240612. Addresses *#5006*.
  - [bdubbs] - Update to bc-6.7.6. Fixes *#5506*.
  - [bdubbs] - Update to man-pages-6.9.1. Fixes *#5507*.
  - [bdubbs] - Update to linux-6.9.7. Fixes *#5508*.
  - [bdubbs] - Update to sysklogd-2.5.2. Fixes *#5509*.

    - [bdubbs] - Update to shadow-4.16.0. Fixes *#5510*.
    - [bdubbs] - Update to setuptools-70.1.1. Fixes *#5512*.
- 2024-06-15
    - [bdubbs] - Update to vim-9.1.0478. Addresses *#4500*.
    - [bdubbs] - Update to iana-etc-20240607. Addresses *#5006*.
    - [bdubbs] - Update to python3-3.12.4. Fixes *#5502*.
    - [bdubbs] - Update to perl-5.40.0. Fixes *#5503*.
    - [bdubbs] - Update to openssl-3.3.1 (Security fix). Fixes *#5500*.
    - [bdubbs] - Update to linux-6.9.4. Fixes *#5505*.
    - [bdubbs] - Update to findutils-4.10.0. Fixes *#5499*.
- 2024-05-31
    - [bdubbs] - Update to meson-1.4.1. Fixes *#5498*.
    - [bdubbs] - Update to xz-5.6.2. Fixes *#5471*.
    - [bdubbs] - Add linux-6.9.x compatibility instructions to udev. Fixes *#5496*.
    - [bdubbs] - Update to setuptools-70.0.0 (python module). Fixes *#5491*.
    - [bdubbs] - Update to ninja-1.12.1. Fixes *#5489*.
    - [bdubbs] - Update to man-pages-6.8. Fixes *#5494*.
    - [bdubbs] - Update to linux-6.9.3. Fixes *#5491*.
    - [bdubbs] - Update to libcap-2.70. Fixes *#5493*.
    - [bdubbs] - Update to iproute2-6.9.0. Fixes *#5492*.
    - [bdubbs] - Update to e2fsprogs-1.47.1. Fixes *#5495*.
- 2024-05-13
    - [xry111] - Synchronize coreutils i18n patch from Fedora to fix a build failure on 32-bit x86 and an alarming compiler warning on both 32-bit x86 and x86_64 with GCC 14.1 or later.
- 2024-05-11
    - [bdubbs] - Update to vim-9.1.0405. Addresses *#4500*.
    - [bdubbs] - Update to util-linux-2.40.1. Fixes *#5482*.
    - [bdubbs] - Update to linux-6.8.9. Fixes *#5484*.
    - [bdubbs] - Update to jinja2-3.1.4 (Python module). Fixes *#5485*.
    - [bdubbs] - Update to iana-etc-20240502. Addresses *#5006*.
    - [bdubbs] - Update to gcc-14.1.0. Fixes *#5486*.
- 2024-05-01
    - [bdubbs] - Add security fix to glibc. Fixes *#5481*.
    - [bdubbs] - Update to linux-6.8.8. Fixes *#5480*.
    - [bdubbs] - Update to ncurses-6.5. Fixes *#5483*.
- 2024-04-16
    - [bdubbs] - Make minor change to ifup script output text.
- 2024-04-15

- [bdubbs] - Update to setuptools-69.5.1. Fixes *#5478*.
- [bdubbs] - Update to python3-3.12.3. Fixes *#5476*.
- [bdubbs] - Update to openssl-3.3.0. Fixes *#5475*.
- [bdubbs] - Update to ninja-1.12.0. Fixes *#5477*.
- [bdubbs] - Update to man-db-2.12.1. Fixes *#5474*.
- [bdubbs] - Update to linux-6.8.6. Fixes *#5472*.
- [bdubbs] - Update to iana-etc-20240412. Addresses *#5006*.
- [bdubbs] - Update to vim-9.1.0330. Addresses *#4500*.

- 2024-03-31
  - [bdubbs] - Update to iana-etc-20240318. Addresses *#5006*.
  - [bdubbs] - Update to zstd-1.5.6. Fixes *#5468*.
  - [bdubbs] - Update to util-linux-2.40. Fixes *#5430*.
  - [bdubbs] - Update to shadow-4.15.1. Fixes *#5465*.
  - [bdubbs] - Update to pkgconf-2.2.0. Fixes *#5469*.
  - [bdubbs] - Update to linux-6.8.2. Fixes *#5467*.
  - [bdubbs] - Update to coreutils-9.5. Fixes *#5470*.
- 2024-03-29
  - [bdubbs] - Revert to xz-5.4.6 due to upstream compromise.
- 2024-03-25
  - [timtas] - Update to sysvinit-3.09. Fixes *#5466*.
- 2024-03-19
  - [renodr] - Update to iproute2-6.8.0. Fixes *#5458*.
  - [renodr] - Update to man-pages-6.7. Fixes *#5464*.
  - [renodr] - Update to Linux-6.8.1. Fixes *#5453*.
  - [renodr] - Added LZ4 to the book. Fixes *#5463*.
- 2024-03-15
  - [bdubbs] - Update to wheel-0.43.0. Fixes *#5459*.
  - [bdubbs] - Update to setuptools-69.2.0 (Python module). Fixes *#5462*.
  - [bdubbs] - Update to meson-1.4.0. Fixes *#5460*.
  - [bdubbs] - Update to expat-2.6.2 (Security fix). Fixes *#5461*.
  - [bdubbs] - Update to iana-etc-20240305. Addresses *#5006*.
  - [bdubbs] - Update to vim-9.1.0161. Addresses *#4500*.
  - [bdubbs] - Update to xz-5.6.1. Fixes *#5457*.
  - [bdubbs] - Update to shadow-4.15.0. Fixes *#5456*.
  - [bdubbs] - Update to psmisc-23.7. Fixes *#5454*.
  - [bdubbs] - Update to kmod-32. Fixes *#5455*.
  - [bdubbs] - Update to elfutils-0.191. Fixes *#5451*.
- 2024-03-02

- [bdubbs] - Update to iana-etc-20240222. Addresses *#5006*.
- [bdubbs] - Update to vim-9.1.0145. Addresses *#4500*.
- [bdubbs] - Update to xz-5.6.0. Fixes *#5447*.
- [bdubbs] - Update to tcl-8.6.14. Fixes *#5448*.
- [bdubbs] - Update to shadow-4.14.6. Fixes *#5450*.
- [bdubbs] - Update to setuptools-69.1.1. Fixes *#5446*.
- [bdubbs] - Update to linux-6.7.7. Fixes *#5444*.
- [bdubbs] - Update to libffi-3.4.6. Fixes *#5443*.
- [bdubbs] - Update to gettext-0.22.5. Fixes *#5445*.
- [bdubbs] - Update to expat-2.6.1. Fixes *#5449*.

- 2024-03-01
  - [bdubbs] - LFS-12.1 released.

1.4. Ресурсы

1.4.1. Часто задаваемые вопросы

Если во время создания системы LFS вы столкнетесь с какими-либо ошибками, у вас возникнут какие-либо вопросы или вам кажется, что в книге допущена опечатка, пожалуйста, для начала ознакомьтесь со списком часто задаваемых вопросов (FAQ), расположенным по адресу *https://mirror.linuxfromscratch.ru/faq/*.

1.4.2. Списки рассылки

На сервере `linuxfromscratch.org` размещен ряд списков рассылки, используемых для разработки проекта LFS. Эти списки включают, среди прочего, основные списки разработки и поддержки. Если вы не можете найти ответ на странице часто задаваемых вопросов, следующим шагом будет поиск решения в списках рассылки по адресу *https://mirror.linuxfromscratch.ru/search.html*.

Для получения информации о списках рассылки, способах подписки, архивах и дополнительной информации посетите *https://mirror.linuxfromscratch.ru/mail.html*.

1.4.3. IRC

Некоторые члены сообщества LFS предлагают помощь в Internet Relay Chat (IRC). Прежде чем воспользоваться этим способом, убедитесь, что на ваш вопрос еще нет ответа в разделе часто задаваемых вопросов LFS или в архивах списков рассылки. Вы можете найти нас в `irc.libera.chat` . Канал поддержки называется #lfs-support.

1.4.4. Зеркала проекта

Проект LFS имеет несколько зеркал по всему миру, чтобы сделать доступ к веб-сайту и загрузку необходимых пакетов более удобными. Пожалуйста, посетите веб-сайт LFS по адресу *https://mirror.linuxfromscratch.ru/mirrors.html* для получения списка текущих зеркал.

1.4.5. Контактная информация

Пожалуйста, направляйте все свои вопросы и комментарии в один из списков рассылки LFS (см. выше).

1.5. Помощь

Примечание

Если вы столкнулись с проблемой при сборки одного пакета с помощью инструкцией из LFS, мы настоятельно не рекомендуем публиковать проблему непосредственно в канале поддержки разработчиков пакета до обсуждения через канал поддержки LFS, указанный в Раздел 1.4, «Ресурсы». Часто это неэффективно, потому что разработчики редко знакомы с процедурой сборки LFS. Даже если вы действительно столкнулись с проблемой в пакете, сообщество LFS все равно может помочь выделить информацию, необходимую специалистам по поддержке пакета, и составить соответствующий отчет.

Если вам нужно задать вопрос напрямую через канал поддержки пакета, вы должны, по крайней мере, понимать, что многие проекты имеют каналы поддержки, отделенные от системы отслеживания ошибок. Отчеты об «ошибках» при отправке вопросов считаются недействительными и могут раздражать разработчиков этих проектов.

Если при работе с этой книгой у вас возникнут проблемы или вопросы, посетите страницу часто задаваемых вопросов по адресу *https://mirror.linuxfromscratch.ru/faq/#generalfaq*. Часто там уже есть ответы на большинство вопросов. Если на этой странице нет ответа на ваш вопрос, попробуйте самостоятельно найти источник проблемы. Следующий документ даст вам некоторые рекомендации по устранению неполадок: *https://mirror.linuxfromscratch.ru/hints/downloads/files/errors.txt*.

Если вы не можете найти решение своей проблемы в разделе часто задаваемых вопросов, выполните поиск в списках рассылки по адресу *https://mirror.linuxfromscratch.ru/search.html*.

У нас также есть замечательное сообщество LFS, которое готово предложить помощь через списки рассылки и IRC (см. Раздел 1.4, «Ресурсы» этой книги). Мы получаем много вопросов в службу поддержки каждый день, и на многие из них можно легко ответить, зайдя в раздел часто задаваемых вопросов и предварительно выполнив поиск в списках рассылки. Чтобы мы могли оказать помощь, вам необходимо сначала провести самостоятельное исследование. Это позволяет нам сосредоточиться на более сложных вопросах в поддержке. Если ваши поиски не привели к решению проблемы, включите всю необходимую информацию (упомянутую ниже) в свой запрос о помощи.

1.5.1. Что следует упомянуть

Помимо краткого объяснения возникшей проблемы, в любой запрос о помощи необходимо включить следующую важную информацию:

- Используемая версия книги (в данном случае 12.2-multilib)
- Информацию о дистрибутиве и его версия, используемые для сборки LFS
- Вывод сценария Системные требования к хостовой машине
- Пакет или раздел где возникла проблема
- Точное сообщение об ошибке или четкое описание проблемы
- Обратите внимание, отклонялись ли вы от книги

Примечание

Отклонение от этой книги *не* означает, что мы не поможем вам. В конце концов, LFS зависит от личных предпочтений. Заблаговременное информирование о любых изменениях в процессе сборки помогает нам оценить и определить возможные причины вашей проблемы

1.5.2. Проблемы со скриптом configure

Если что-то пойдет не так во время выполнения скрипта **configure**, просмотрите файл `config.log` . Этот файл может содержать ошибки, обнаруженные во время настройки, которые не были выведены на экран. Включите *соответствующие* строки, если вам нужно обратиться за помощью.

1.5.3. Проблемы компиляции

Как вывод на экран, так и содержимое различных файлов полезны для определения причины проблем компиляции. Вывод экрана из скрипта **configure** и запуска **make** может быть полезен. Необязательно включать весь вывод целиком, но обязательно включите всю необходимую информацию. Ниже приведен пример информации, которая должна быть включена в экранный вывод **make**:

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

В этом случае многие люди просто включили бы только нижнюю часть:

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

Этой информации недостаточно, чтобы правильно диагностировать проблему, потому что она только указывает на то, что что-то пошло не так, а не на то, *что* пошло не так. Весь раздел, как в приведенном выше примере, должен быть сохранен, так как он включает в себя выполненную команду и все связанные с ней сообщения об ошибках.

Отличная статья о том, как обращаться за помощью в Интернете, доступна по адресу *http://catb.org/~esr/faqs/smart-questions.html*. Прочтите этот документ и следуйте советам, чтобы повысить вероятность получения помощи в которой вы нуждаетесь.

Часть II. Подготовка к сборке

Глава 2. Подготовка хост-системы

2.1. Введение

В этой главе проверяются и при необходимости устанавливаются основные инструменты, необходимые для построения LFS. Затем подготавливается раздел, в котором будет размещаться система LFS. Мы создадим сам раздел, создадим на нем файловую систему и смонтируем его.

2.2. Требования к хост-системе

2.2.1. Аппаратное обеспечение

Редакторы LFS рекомендуют, чтобы процессор имел не менее четырех ядер и не менее 8 ГБ памяти. Старые системы, не отвечающие этим требованиям, будут по-прежнему работать, но время сборки пакетов будет значительно больше, чем указано в документации.

2.2.2. Программное обеспечение

Ваша хост-система должна иметь следующее программное обеспечение с указанными минимальными версиями. Это не должно быть проблемой для большинства современных дистрибутивов Linux. Также обратите внимание на то, что многие дистрибутивы помещают заголовочные файлы в отдельные пакеты, как правило в формате `<package-name>-devel` или `<package-name>-dev`. Обязательно установите эти пакеты, если ваш дистрибутив их предоставляет.

Более ранние версии перечисленных ниже пакетов могут работать, но это не проверялось.

- **Bash-3.2** (/bin/sh должен быть символической или жесткой ссылкой на bash)
- **Binutils-2.13.1** (Версия выше 2.43.1 не рекомендуется, так как она не тестировалась)
- **Bison-2.7** (/usr/bin/yacc должен быть ссылкой на bison или небольшой скрипт, запускающий bison)
- **Coreutils-8.1**
- **Diffutils-2.8.1**
- **Findutils-4.2.31**
- **Gawk-4.0.1** (/usr/bin/awk должен быть ссылкой на gawk)
- **GCC-5.2**, включая компилятор C++, **g++** (версии выше 14.2.0 не рекомендуются, поскольку они не тестировались). Также должны присутствовать стандартные библиотеки C и C++ (с заголовочными файлами), чтобы компилятор C++ мог осуществлять сборку программ.
- **Grep-2.5.1a**
- **Gzip-1.3.12**
- **Linux Kernel-4.19**

  Причиной, по которой указаны минимальные требования к версии ядра, является то, что мы указываем эту версию при сборке glibc в Глава 5 и Глава 8. Так как более старые ядра не поддерживаются, скомпилированный пакет glibc немного меньше и быстрее. По состоянию на февраль 2024 г. 4.19 является самой старой версией ядра, поддерживаемой разработчиками ядра. Некоторые версии ядра, более старые, чем 4.19, могут по-прежнему поддерживаться сторонними командами, но они не считаются официальными выпусками ядра; подробности читайте на странице *https://kernel.org/category/releases.html*

  Если версия ядра хоста более ранняя, чем 4.19, вам необходимо обновить ядро на более современную версию. Есть два способа сделать это. Во-первых, посмотрите, предоставляет ли ваш дистрибутив Linux пакет ядра 4.19 или более позднюю версию. Если это так, установите его. Если ваш дистрибутив не предлагает приемлемый пакет ядра или вы предпочитаете не устанавливать его, вы можете скомпилировать ядро самостоятельно. Инструкции по компиляции ядра и настройке загрузчика (при условии, что хост использует GRUB) находятся в Глава 10.

Для сборки LFS необходимо, чтобы ядро хоста поддерживало псевдотерминал UNIX 98 (PTY). Обычно он включен на всех настольных или серверных дистрибутивах, поставляющих Linux 4.19 или более новое ядро. Если на хосте вы используете самостоятельно собранное ядро, убедитесь, что для параметра `CONFIG_UNIX98_PTYS` установлено значение `y` в конфигурационном файле ядра.

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

Важно

Обратите внимание, что упомянутые выше символические ссылки необходимы для создания системы LFS с использованием инструкций, содержащихся в этой книге. Симлинки, указывающие на другое программное обеспечение (например, dash, mawk и т. д.), могут работать, но не тестируются и не поддерживаются командой разработчиков LFS, и могут потребовать либо отклонения от инструкций, либо дополнительных исправлений для некоторых пакетов.

Чтобы узнать, есть ли в вашей хост-системе все необходимые пакеты и возможность компилировать программы, выполните следующий скрипт:

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

# If you have tools installed in other directories, adjust PATH here AND
# in ~lfs/.bashrc (section 4.4) as well.

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

bail() { echo "FATAL: $1"; exit 1; }
grep --version > /dev/null 2> /dev/null || bail "grep does not work"
sed '' /dev/null || bail "sed does not work"
sort   /dev/null || bail "sort does not work"

ver_check()
{
   if ! type -p $2 &>/dev/null
   then
     echo "ERROR: Cannot find $2 ($1)"; return 1;
   fi
   v=$($2 --version 2>&1 | grep -E -o '[0-9]+\.[0-9\.]+[a-z]*' | head -n1)
   if printf '%s\n' $3 $v | sort --version-sort --check &>/dev/null
   then
     printf "OK:    %-9s %-6s >= $3\n" "$1" "$v"; return 0;
   else
     printf "ERROR: %-9s is TOO OLD ($3 or later required)\n" "$1";
     return 1;
   fi
}

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

```
   then
     printf "OK:    Linux Kernel $kver >= $1\n"; return 0;
   else
     printf "ERROR: Linux Kernel ($kver) is TOO OLD ($1 or later required)\n" "$kver";
     return 1;
   fi
}

# Coreutils first because --version-sort needs Coreutils >= 7.0
ver_check Coreutils      sort     8.1 || bail "Coreutils too old, stop"
ver_check Bash           bash     3.2
ver_check Binutils       ld       2.13.1
ver_check Bison          bison    2.7
ver_check Diffutils      diff     2.8.1
ver_check Findutils      find     4.2.31
ver_check Gawk           gawk     4.0.1
ver_check GCC            gcc      5.2
ver_check "GCC (C++)"    g++      5.2
ver_check Grep           grep     2.5.1a
ver_check Gzip           gzip     1.3.12
ver_check M4             m4       1.4.10
ver_check Make           make     4.0
ver_check Patch          patch    2.5.4
ver_check Perl           perl     5.8.8
ver_check Python         python3  3.4
ver_check Sed            sed      4.1.5
ver_check Tar            tar      1.22
ver_check Texinfo        texi2any 5.0
ver_check Xz             xz       5.0.0
ver_kernel 4.19

if mount | grep -q 'devpts on /dev/pts' && [ -e /dev/ptmx ]
then echo "OK:    Linux Kernel supports UNIX 98 PTY";
else echo "ERROR: Linux Kernel does NOT support UNIX 98 PTY"; fi

alias_check() {
   if $1 --version 2>&1 | grep -qi $2
   then printf "OK:    %-4s is $2\n" "$1";
   else printf "ERROR: %-4s is NOT $2\n" "$1"; fi
}
echo "Aliases:"
alias_check awk GNU
alias_check yacc Bison
alias_check sh Bash

echo "Compiler check:"
if printf "int main(){}" | g++ -x c++ -
then echo "OK:    g++ works";
else echo "ERROR: g++ does NOT work"; fi
rm -f a.out

if [ "$(nproc)" = "" ]; then
   echo "ERROR: nproc is not available or it produces empty output"
else
   echo "OK: nproc reports $(nproc) logical cores are available"
fi
EOF

bash version-check.sh
```

Для сборки multilib-системы требуется, чтобы в ядре хост-системы была включена поддержка 32-разрядной эмуляции.

```
Binary Emulations  --->
    [*] IA32 Emulation
    <M>   IA32 a.out support
    [*] x32 ABI for 64-bit mode
```

Опция 'IA32 a.out support' является необязательной. Если в вашем ядре не включена опция 'x32 ABI for 64-bit mode', но включена 'IA32 Emulation', вы можете продолжить сборку своей системы, однако, вам придется исключить любые разделы, содержащие инструкции по сборке объектов x32. Если ни 'IA32 Emulation', ни 'x32 ABI for 64-bit mode' не включены, будут возникать ошибки при сборке glibc в главе 6, поэтому необходимо обновление ядра вашей хост-системы.

2.3. Этапы сборки системы LFS

LFS разработан для сборки за один сеанс. То есть инструкция предполагает, что система не будет выключаться в процессе. Это не означает, что система должна быть собрана за один присест. Для возобновления сборки в точке предыдущей остановки (после перезагрузки/выключения), необходимо выполнить некоторые процедуры повторно.

2.3.1. Главы 1–4

Эти главы выполняются на хост-системе. После перезагрузки обратите внимание на следующее:

- При выполнении операций, от имени пользователя `root` после Раздела 2.4, *ДЛЯ ПОЛЬЗОВАТЕЛЯ* `root` должна быть установлена переменная окружения LFS.

2.3.2. Главы 5–6

- Раздел /mnt/lfs должен быть смонтирован.
- Эти две главы *должны* быть выполнены из-под пользователя `lfs`. Перед выполнением любой задачи в этих главах необходимо выполнить команду **su - lfs**. В противном случае вы рискуете установить пакеты на хост и сделать его непригодным для использования.
- Выполнение процедур из Общие инструкции по компиляции имеет решающее значение. Если есть какие-либо сомнения по поводу установки пакета, убедитесь, что все ранее распакованные tar-архивы удалены, затем повторно извлеките файлы и выполните все инструкции, приведенные в этом разделе.

2.3.3. Главы 7–10

- Раздел /mnt/lfs должен быть смонтирован.
- Некоторые операции, такие как «Смена владельца» или «Вход в среду Chroot», должны быть выполнены от имени пользователя `root` с переменной окружения $LFS, установленной для пользователя `root`.
- При входе в chroot переменная среды LFS должна быть установлена для пользователя `root`. Переменная LFS не используется после входа в среду chroot.
- Виртуальные файловые системы должны быть смонтированы. Это можно сделать до или после входа в chroot, переключившись на виртуальный терминал хоста и от имени пользователя `root` выполнив команды, описанные в Раздел 7.3.1, «Монтирование и заполнение /dev» и Раздел 7.3.2, «Монтирование виртуальных файловых систем ядра».

2.4. Создание нового раздела

Как и большинство других операционных систем, LFS обычно устанавливается на выделенный раздел. Рекомендуемый подход к построению системы LFS состоит в том, чтобы использовать доступный пустой раздел или, если у вас достаточно неразмеченного пространства, использовать его

Минимальная система требует раздел размером около 10 гигабайт (ГБ). Этого достаточно для хранения всех архивов с исходным кодом и компиляции пакетов. Однако, если система LFS предназначена для использования в качестве основной системы Linux, вероятно, будет установлено дополнительное программное обеспечение, для которого потребуется дополнительное пространство. Раздел размером 30 ГБ является разумным размером для расширения. Сама система LFS не займет столько места. Большая

часть этого требования заключается в предоставлении достаточного временного хранилища, а также в добавлении дополнительных возможностей после сборки LFS. Кроме того, для компиляции пакетов может потребоваться много места на диске, которое будет освобождено после установки пакета.

Поскольку для компиляции не всегда достаточно оперативной памяти (ОЗУ), рекомендуется использовать небольшой раздел диска в качестве раздела подкачки. Он используется ядром для хранения редко используемых данных и оставляет больше памяти для активных процессов. Раздел подкачки для системы LFS может совпадать с разделом, используемым хост-системой, и в этом случае нет необходимости создавать еще один.

Запустите программу создания разделов диска, такую как **cfdisk** или **fdisk**, с параметром командной строки, указав имя жесткого диска, на котором будет создан новый раздел, например, /dev/sda для основного диска. Создайте раздел Linux и раздел подкачки, если это необходимо. Пожалуйста, обратитесь к справке *cfdisk(8)* или *fdisk(8)*, если вы еще не знаете, как пользоваться этими программами.

Примечание

Для опытных пользователей возможны и другие схемы разбиения. Система LFS может располагаться на программном *RAID-массиве* или логическом томе *LVM*. Однако для некоторых опций требуется *initramfs*, что является сложной темой. Эти методы разбиения не рекомендуются начинающим пользователям LFS.

Запомните обозначение созданного раздела (например, sda5). В этой книге он будет называться разделом LFS. Также запомните обозначение раздела подкачки. Эти имена понадобятся позже для файла /etc/fstab .

2.4.1. Другие вопросы по созданию разделов

Рекомендации по созданию разделов системы часто публикуются в списках рассылки LFS. Это очень субъективная тема. По умолчанию для большинства дистрибутивов используется весь диск, за исключением небольшого раздела подкачки. Это не оптимально для LFS по нескольким причинам. Это снижает гибкость, затрудняет совместное использование данных между несколькими дистрибутивами или сборками LFS, делает резервное копирование более трудоемким и может тратить дисковое пространство из-за неэффективно распределенной файловой системы.

2.4.1.1. Корневой раздел

Корневой раздел LFS (не путать с каталогом /root) размером в 20 гигабайт является хорошим компромиссом для большинства систем. Он обеспечивает достаточно места для построения LFS и большей части BLFS, но достаточно мал, чтобы можно было легко создать несколько разделов для экспериментов.

2.4.1.2. Раздел подкачки

Большинство дистрибутивов автоматически создают раздел подкачки. Обычно рекомендуемый размер раздела подкачки примерно в два раза превышает объем физической памяти, однако это требуется редко. Если дисковое пространство ограничено, установите размер раздела подкачки в два гигабайта и контролируйте его объемом.

Если вы хотите использовать режим гибернации (suspend-to-disk) Linux, которая записывает содержимое ОЗУ в раздел подкачки перед выключением машины. Установите размер раздела подкачки не меньше объема установленной оперативной памяти.

Использование файла подкачки - это не очень хорошо. Для механических жестких дисков вы можете определить, что система использует раздел подкачки, просто слыша активность диска и наблюдая, как система реагирует на команды. Для SSD-накопителя вы не сможете услышать, что используется раздел подкачки, но сможете оценить, сколько места на разделе подкачки занято, используя команды **top** или **free**. По возможности следует избегать использования SSD-накопителя для раздела подкачки. Первой

реакцией на активность раздела подкачки должна быть проверка на необоснованное применение какой-либо команды, например, попытка редактирования пятигигабайтного файла. Если использование раздела подкачки становится обычным явлением, лучшее решение — приобретение большего объема оперативной памяти для вашей системы.

2.4.1.3. Раздел GRUB

Если *загрузочный диск* размечен с помощью таблицы разделов GUID (GPT), необходимо создать небольшой раздел, обычно размером 1 МБ, если он еще не существует. Этот раздел не форматируется, но должен быть доступен для использования GRUB во время установки загрузчика. Обычно он помечен как 'BIOS Boot' при использовании **fdisk** или имеет код *EF02* при использовании **gdisk**.

Примечание

Раздел Grub Bios должен находиться на диске, который BIOS использует для загрузки системы. Это не обязательно тот же диск, на котором расположен корневой раздел LFS. Диски в системе могут использовать разные типы таблиц разделов. Наличие раздела Grub Bios зависит только от типа таблицы разделов на загрузочном диске.

2.4.1.4. Разделы, используемые для удобства

Есть несколько других разделов, которые не являются обязательными, но их следует учитывать при разработке схемы диска. Следующий список не является исчерпывающим, а представлен в качестве справочного руководства.

- /boot – Настоятельно рекомендуется. Используйте этот раздел для хранения ядер и другой загрузочной информации. Чтобы свести к минимуму возможные проблемы с загрузкой дисков большого размера, сделайте этот раздел первым физическим разделом на первом диске. Размер раздела в 200 мегабайт вполне достаточен.
- /boot/efi – Системный раздел EFI, используемый для загрузки системы с помощью UEFI. Подробнее читайте на *странице BLFS*.
- /home – Настоятельно рекомендуется. Предоставьте общий доступ к своему домашнему каталогу и пользовательским настройкам нескольким дистрибутивам или сборкам LFS. Размер, как правило, довольно большой и зависит от доступного места на диске.
- /usr – в LFS, `/bin` , `/lib` , и `/sbin` являются символическими ссылками на их аналоги в `/usr` . Таким образом `/usr` содержит все двоичные файлы, необходимые для работы системы. Для LFS отдельный раздел `/usr` не требуется. Если он вам необходим, вы должны сделать раздел достаточно большим, чтобы поместить туда все программы и библиотеки в системе. В этой конфигурации, корневой раздел может быть очень маленьким (возможно, всего один гигабайт), поэтому он подходит для тонкого клиента или бездисковой рабочей станции (где `/usr` монтируется с удаленного сервера). Однако вы должны знать, что для загрузки системы с отдельного раздела `/usr` потребуется initramfs (не включенный в LFS).
- /opt – Этот каталог наиболее полезен для BLFS, в него можно установить некоторые большие пакеты, такие как KDE или Texlive, без использования иерархии /usr. Для /opt достаточно размера от 5 до 10 гигабайт.
- /tmp – Отдельный раздел /tmp используется редко, он полезен при настройке тонкого клиента. Обычно, его размер не должен превышать пару гигабайт. Если у вас достаточно оперативной памяти, вы можете смонтировать /tmp как `tmpfs`, чтобы ускорить доступ к временным файлам.
- /usr/src – Этот раздел очень удобен для хранения исходников BLFS и совместного использования их в сборках LFS. Его также можно использовать в качестве места для сборки пакетов BLFS. Размера в 30-50 гигабайт вполне достаточно.

Любой отдельный раздел, который вы хотите автоматически монтировать при загрузке, должен быть указан в файле `/etc/fstab` . Подробности о том, как указать разделы, будут обсуждаться в Раздел 10.2, «Создание файла /etc/fstab».

2.5. Создание файловой системы на разделе

Раздел - это всего лишь диапазон секторов на диске, указанный в таблице разделов. Прежде чем операционная система сможет использовать раздел для хранения каких-либо файлов, он должен быть отформатирован, чтобы содержать файловую систему, обычно состоящую из метки, блоков каталогов, блоков данных и схемы индексации для поиска конкретного файла по запросу. Файловая система также помогает операционной системе отслеживать свободное пространство на разделе, резервировать необходимые секторы при создании нового файла или расширении существующего и повторно использует свободные сегменты данных, полученные в результате удаления файлов. Она также может обеспечивать поддержку избыточности данных и восстановления после ошибок.

LFS может использовать любую файловую систему, распознаваемую ядром Linux, но наиболее распространенными типами являются ext3 и ext4. Выбор правильной файловой системы может быть сложным; это зависит от характеристик файлов и размера раздела. Например:

ext2
: подходит для небольших разделов, которые редко обновляются, например /boot.

ext3
: это обновленная файловая система ext2, которая включает в себя журнал, помогающий восстановить состояние раздела в случае некорректного завершения работы. Обычно используется в качестве файловой системы общего назначения.

ext4
: является последней версией файловых систем семейства ext. Она предоставляет несколько новых возможностей, включая временные метки с точностью до наносекунды, создание и использование очень больших файлов (16 ТБ) и повышение скорости работы.

Другие файловые системы, включая FAT32, NTFS, JFS и XFS, полезны для конкретных задач. Более подробную информацию об этих файловых системах и многих других можно найти по адресу *https://en.wikipedia.org/wiki/Comparison_of_file_systems*.

LFS предполагает, что корневая файловая система (/) имеет тип ext4. Чтобы создать файловую систему `ext4` на разделе LFS, выполните следующую команду:

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

Замените *<xxx>* именем раздела LFS

Если вы используете существующий `раздел подкачки`, нет необходимости его форматировать. Если был создан новый `раздел подкачки`, его нужно будет инициализировать с помощью этой команды:

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

Замените *<yyy>* именем `раздела подкачки`.

2.6. Установка переменной $LFS

В этой книге переменная окружения `LFS` будет использоваться несколько раз. Вы должны убедиться, что эта переменная всегда определена в процессе сборки LFS. Она должна быть установлена на каталог, в котором вы будете создавать свою систему LFS — мы, для примера, будем использовать `/mnt/lfs` , но вы можете выбрать любой другой. Если вы собираете LFS на отдельном разделе, этот каталог будет точкой монтирования для раздела. Выберите расположение каталога и установите переменную с помощью следующей команды:

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

Установка этой переменной полезна тем, что такие команды, как **mkdir -v $LFS/tools**, можно вводить буквально. Оболочка автоматически заменит «$LFS» на «/mnt/lfs» (или любое другое значение переменной) при обработке команды.

Внимание

Не забывайте проверять, что переменная LFS установлена, всякий раз, когда вы покидаете и снова входите в текущую рабочую среду (например, когда выполняете **su** для root или другого пользователя). Убедитесь, что переменная LFS настроена правильно:

```
echo $LFS
```

Убедитесь, что в выходных данных указан путь к местоположению сборки вашей системы LFS, то есть /mnt/lfs , если вы следовали примеру. Если вывод неверен, используйте команду, указанную ранее, чтобы установить $LFS в правильное значение каталога LFS.

Примечание

Один из способов гарантировать, что переменная LFS всегда установлена, — отредактировать файл .bash_profile как в вашем личном домашнем каталоге, так и в /root/.bash_profile и добавить приведенную выше команду экспорта. Кроме того, оболочка, указанная в файле /etc/passwd для всех пользователей, которым нужна переменная LFS, должна быть bash, чтобы гарантировать, что файл /root/.bash_profile используется как часть процесса входа в систему.

Еще один способ, который используется для входа в хост-систему. При входе в систему через диспетчер графического дисплея пользовательский .bash_profile не используется при запуске виртуального терминала. В этом случае добавьте команду экспорта в файл .bashrc для своего пользователя и root. Кроме того, некоторые дистрибутивы используют тест "if" и не запускают оставшиеся инструкции .bashrc для не интерактивного вызова bash. Обязательно разместите команду экспорта перед тестом для не интерактивного использования.

2.7. Монтирование нового раздела

Теперь, когда файловая система создана, раздел должен быть смонтирован, чтобы хост-система могла получить доступ к нему. В книге предполагается, что файловая система монтируется в каталог, указанный в переменной LFS, описанной в предыдущем разделе.

Строго говоря, нельзя «смонтировать раздел». Монтируется *файловая система* на этом разделе. Но так как один раздел не может содержать несколько файловых систем, люди часто говорят о разделе и связанной с ним файловой системе так, как если бы они были одним и тем же.

Создайте точку монтирования и смонтируйте файловую систему LFS с помощью этих команд:

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

Замените *<xxx>* на имя раздела LFS.

Если вы используете несколько разделов для LFS (например, один для /, а другой для /home), смонтируйте их вот так:

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

Замените *<xxx>* и *<yyy>* соответствующими именами разделов.

Убедитесь, что этот новый раздел не смонтирован со слишком строгими разрешениями (такими как параметры `nosuid` или `nodev`). Запустите команду **mount** без каких-либо параметров, чтобы увидеть, какие параметры установлены для смонтированного раздела LFS. Если установлены `nosuid` и/или `nodev`, раздел должен быть размонтирован и смонтирован повторно.

Предупреждение

Приведенные выше инструкции предполагают, что вы не будете перезагружать компьютер в процессе сборки LFS. Если вы выключите свою систему, вам придется либо перемонтировать раздел LFS каждый раз, когда вы перезапускаете процесс сборки, либо изменить файл `/etc/fstab` вашей хост-системы, чтобы он автоматически монтировал его при загрузке. Например, вы можете добавить эту строку в свой `/etc/fstab` :

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

Если вы используете дополнительные разделы, обязательно добавьте их.

Если вы используете `раздел подкачки`, убедитесь, что он включен с помощью команды **swapon**:

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

Замените `<zzz>` именем `раздела подкачки`.

Теперь, когда новый раздел LFS готов к работе, пришло время загрузить пакеты.

Глава 3. Пакеты и патчи

3.1. Введение

Эта глава содержит список пакетов, которые необходимо загрузить для сборки базовой системы Linux. Перечисленные версии программного обеспечения, соответствуют версиям, которые, проверены и работают, книга основана на их использовании. Мы настоятельно рекомендуем не использовать другие версии пакетов, потому что команды сборки для одной версии могут не работать с другой, если только другая версия не указана в сообщениях об ошибках LFS или рекомендациях по безопасности. В новейших версиях пакетов также могут быть проблемы, требующие обходных путей. Эти обходные пути будут стабилизированы в разрабатываемой версии книги.

Для некоторых пакетов архив релиза и архив снимка репозитория (Git или SVN) для этого выпуска могут быть опубликованы с одинаковыми именами файлов. Релиз содержит сгенерированные файлы (например, скрипт **configure**, сгенерированный пакетом **autoconf**) в дополнение к содержимому соответствующего моментального снимка репозитория. В книге везде, где это возможно, используются релизные архивы. Использование моментального снимка вместо tar-архива, указанного в книге, может вызвать проблемы.

Источники загрузки могут быть недоступны. Если источник изменился с момента публикации этой книги, Google (*https://www.google.com/*) предоставляет удобную поисковую систему для поиска большинства пакетов. Если поиск не увенчался успехом, попробуйте один из альтернативных способов загрузки, расположенных по адресу *https://mirror.linuxfromscratch.ru/lfs/mirrors.html#files*.

Загруженные пакеты и патчи необходимо где-нибудь хранить, чтобы они были доступны на протяжении всей сборки. Рабочий каталог также необходим для распаковки исходников и их сборки. `$LFS/sources` можно использовать и как место для хранения архивов и патчей, и как рабочий каталог. При использовании этого каталога необходимые элементы будут расположены в разделе LFS и будут доступны на всех этапах процесса сборки.

Чтобы создать этот каталог, выполните следующую команду от имени пользователя `root` перед началом загрузки:

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

Сделайте этот каталог доступным для записи и установите sticky bit (дословно "липкий бит"). Атрибут «Sticky bit» означает, что даже если несколько пользователей имеют право на запись в каталог, только владелец файла может удалить файл в таком каталоге. Следующая команда активирует режимы записи и sticky bit:

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

Есть несколько способов получить все необходимые пакеты и патчи для сборки LFS:

- Файлы можно загрузить по отдельности, как описано в следующих двух разделах.
- Для стабильных версий книги архив со всеми необходимыми файлами можно загрузить с одного из зеркал LFS, перечисленных на странице *https://mirror.linuxfromscratch.ru/mirrors.html#files*.
- Файлы можно загрузить с помощью **wget** и wget-list.

Чтобы загрузить все пакеты и патчи, используя *wget-list-sysv* в качестве входных данных для команды wget, наберите команду:

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

Начиная с LFS-7.0, существует отдельный файл *md5sums*, который можно использовать для проверки всех пакетов. Поместите этот файл в `$LFS/sources` и выполните:

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

Эту проверку можно использовать после загрузки файлов любым из перечисленных выше способов.

Если пакеты и исправления загружаются от имени пользователя, без привилегий `root`, то файлы будут принадлежать этому пользователю. Файловая система записывает владельца по его UID, а UID обычного пользователя в хост-дистрибутиве не будет присвоен в LFS. Таким образом, файлы останутся принадлежащими безымянному UID в конечной системе LFS. Если вы не назначили тот же UID для своего пользователя в системе LFS, измените владельца этих файлов на `root` сейчас, чтобы избежать этой проблемы:

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

3.2. Все пакеты

Примечание

Ознакомьтесь с *рекомендациями по безопасности* перед загрузкой пакетов, чтобы узнать, следует ли использовать более новую версию пакета, чтобы избежать проблем безопасности.

При выходе новых версий, старые версии пакетов могут быть удалены, особенно, если они содержали уязвимости. Если одна или несколько ссылок ниже недоступны, сначала ознакомьтесь с рекомендациями по безопасности, чтобы понять следует ли использовать более новую версию (с исправленной уязвимостью). Если нет, попробуйте скачать удаленный пакет с зеркала. Хотя старый релиз можно скачать с зеркала (даже если он был удален из-за уязвимости), для сборки системы не рекомендуется использовать версию, которая уязвима.

Загрузите или иным образом получите следующие пакеты:

• Acl (2.3.2) - 363 KB:
Домашняя страница: *https://savannah.nongnu.org/projects/acl*
Ссылка на загрузку: *https://download.savannah.gnu.org/releases/acl/acl-2.3.2.tar.xz*
Контрольная сумма MD5: `590765dee95907dbc3c856f7255bd669`

• Attr (2.5.2) - 484 KB:
Домашняя страница: *https://savannah.nongnu.org/projects/attr*
Ссылка на загрузку: *https://download.savannah.gnu.org/releases/attr/attr-2.5.2.tar.gz*
Контрольная сумма MD5: `227043ec2f6ca03c0948df5517f9c927`

• Autoconf (2.72) - 1,360 KB:
Домашняя страница: *https://www.gnu.org/software/autoconf/*
Ссылка на загрузку: *https://ftp.gnu.org/gnu/autoconf/autoconf-2.72.tar.xz*
Контрольная сумма MD5: `1be79f7106ab6767f18391c5e22be701`

• Automake (1.17) - 1,614 KB:
Домашняя страница: *https://www.gnu.org/software/automake/*
Ссылка на загрузку: *https://ftp.gnu.org/gnu/automake/automake-1.17.tar.xz*
Контрольная сумма MD5: `7ab3a02318fee6f5bd42adfc369abf10`

• Bash (5.2.32) - 10,697 KB:
Домашняя страница: *https://www.gnu.org/software/bash/*
Ссылка на загрузку: *https://ftp.gnu.org/gnu/bash/bash-5.2.32.tar.gz*
Контрольная сумма MD5: `f204835b2e06c06e37b5ad776ff907f4`

• Bc (6.7.6) - 463 KB:
Домашняя страница: *https://git.gavinhoward.com/gavin/bc*
Ссылка на загрузку: *https://github.com/gavinhoward/bc/releases/download/6.7.6/bc-6.7.6.tar.xz*
Контрольная сумма MD5: `a47aa5e4e7395fbcd159a9228613b97b`

• Binutils (2.43.1) - 27,514 KB:
Домашняя страница: *https://www.gnu.org/software/binutils/*
Ссылка на загрузку: *https://sourceware.org/pub/binutils/releases/binutils-2.43.1.tar.xz*
Контрольная сумма MD5: `9202d02925c30969d1917e4bad5a2320`

• Bison (3.8.2) - 2,752 KB:
Домашняя страница: *https://www.gnu.org/software/bison/*
Ссылка на загрузку: *https://ftp.gnu.org/gnu/bison/bison-3.8.2.tar.xz*
Контрольная сумма MD5: `c28f119f405a2304ff0a7ccdcc629713`

• Bzip2 (1.0.8) - 792 KB:
Ссылка на загрузку: *https://www.sourceware.org/pub/bzip2/bzip2-1.0.8.tar.gz*
Контрольная сумма MD5: `67e051268d0c475ea773822f7500d0e5`

• Check (0.15.2) - 760 KB:
Домашняя страница: *https://libcheck.github.io/check*
Ссылка на загрузку: *https://github.com/libcheck/check/releases/download/0.15.2/check-0.15.2.tar.gz*
Контрольная сумма MD5: `50fcafcecde5a380415b12e9c574e0b2`

• Coreutils (9.5) - 5,867 KB:
Домашняя страница: *https://www.gnu.org/software/coreutils/*
Ссылка на загрузку: *https://ftp.gnu.org/gnu/coreutils/coreutils-9.5.tar.xz*
Контрольная сумма MD5: `e99adfa059a63db3503cc71f3d151e31`

• DejaGNU (1.6.3) - 608 KB:
Домашняя страница: *https://www.gnu.org/software/dejagnu/*
Ссылка на загрузку: *https://ftp.gnu.org/gnu/dejagnu/dejagnu-1.6.3.tar.gz*
Контрольная сумма MD5: `68c5208c58236eba447d7d6d1326b821`

• Diffutils (3.10) - 1,587 KB:
Домашняя страница: *https://www.gnu.org/software/diffutils/*
Ссылка на загрузку: *https://ftp.gnu.org/gnu/diffutils/diffutils-3.10.tar.xz*
Контрольная сумма MD5: `2745c50f6f4e395e7b7d52f902d075bf`

• E2fsprogs (1.47.1) - 9,720 KB:
Домашняя страница: *https://e2fsprogs.sourceforge.net/*
Ссылка на загрузку: *https://downloads.sourceforge.net/project/e2fsprogs/e2fsprogs/v1.47.1/e2fsprogs-1.47.1.tar.gz*
Контрольная сумма MD5: `75e6d1353cbe6d5728a98fb0267206cb`

• Elfutils (0.191) - 9,092 KB:
Домашняя страница: *https://sourceware.org/elfutils/*
Ссылка на загрузку: *https://sourceware.org/ftp/elfutils/0.191/elfutils-0.191.tar.bz2*
Контрольная сумма MD5: `636547248fb3fae58ec48030298d3ef7`

• Expat (2.6.2) - 474 KB:
Домашняя страница: *https://libexpat.github.io/*
Ссылка на загрузку: *https://prdownloads.sourceforge.net/expat/expat-2.6.2.tar.xz*
Контрольная сумма MD5: `0cb75c8feb842c0794ba89666b762a2d`

• Expect (5.45.4) - 618 KB:
Домашняя страница: *https://core.tcl.tk/expect/*
Ссылка на загрузку: *https://prdownloads.sourceforge.net/expect/expect5.45.4.tar.gz*
Контрольная сумма MD5: `00fce8de158422f5ccd2666512329bd2`

• File (5.45) - 1,218 KB:
Домашняя страница: *https://www.darwinsys.com/file/*
Ссылка на загрузку: *https://astron.com/pub/file/file-5.45.tar.gz*
Контрольная сумма MD5: `26b2a96d4e3a8938827a1e572afd527a`

• **Findutils (4.10.0) - 2,189 KB:**
Домашняя страница: *https://www.gnu.org/software/findutils/*
Ссылка на загрузку: *https://ftp.gnu.org/gnu/findutils/findutils-4.10.0.tar.xz*
Контрольная сумма MD5: `870cfd71c07d37ebe56f9f4aaf4ad872`

• **Flex (2.6.4) - 1,386 KB:**
Домашняя страница: *https://github.com/westes/flex*
Ссылка на загрузку: *https://github.com/westes/flex/releases/download/v2.6.4/flex-2.6.4.tar.gz*
Контрольная сумма MD5: `2882e3179748cc9f9c23ec593d6adc8d`

• **Flit-core (3.9.0) - 41 KB:**
Домашняя страница: *https://pypi.org/project/flit-core/*
Ссылка на загрузку: *https://pypi.org/packages/source/f/flit-core/flit_core-3.9.0.tar.gz*
Контрольная сумма MD5: `3bc52f1952b9a78361114147da63c35b`

• **Gawk (5.3.0) - 3,356 KB:**
Домашняя страница: *https://www.gnu.org/software/gawk/*
Ссылка на загрузку: *https://ftp.gnu.org/gnu/gawk/gawk-5.3.0.tar.xz*
Контрольная сумма MD5: `97c5a7d83f91a7e1b2035ebbe6ac7abd`

• **GCC (14.2.0) - 90,144 KB:**
Домашняя страница: *https://gcc.gnu.org/*
Ссылка на загрузку: *https://ftp.gnu.org/gnu/gcc/gcc-14.2.0/gcc-14.2.0.tar.xz*
Контрольная сумма MD5: `2268420ba02dc01821960e274711bde0`

• **GDBM (1.24) - 1,168 KB:**
Домашняя страница: *https://www.gnu.org/software/gdbm/*
Ссылка на загрузку: *https://ftp.gnu.org/gnu/gdbm/gdbm-1.24.tar.gz*
Контрольная сумма MD5: `c780815649e52317be48331c1773e987`

• **Gettext (0.22.5) - 10,031 KB:**
Домашняя страница: *https://www.gnu.org/software/gettext/*
Ссылка на загрузку: *https://ftp.gnu.org/gnu/gettext/gettext-0.22.5.tar.xz*
Контрольная сумма MD5: `3ae5580599d84be93e6213930facb2db`

• **Glibc (2.40) - 18,313 KB:**
Домашняя страница: *https://www.gnu.org/software/libc/*
Ссылка на загрузку: *https://ftp.gnu.org/gnu/glibc/glibc-2.40.tar.xz*
Контрольная сумма MD5: `b390feef233022114950317f10c4fa97`

Примечание

Разработчики Glibc поддерживают *Git ветку* содержащую исправления, которые заслуживают внимания для Glibc-2.40 но, к сожалению, выпущенные после релиза Glibc-2.40. Редакторы LFS публикуют рекомендации по безопасности, если в ветку добавлено какое-либо исправление безопасности, но для других недавно добавленных патчей не будет предпринято никаких действий. Вы можете самостоятельно просмотреть патчи и включить некоторые из них, если посчитаете их важными.

• **GMP (6.3.0) - 2,046 KB:**
Домашняя страница: *https://www.gnu.org/software/gmp/*
Ссылка на загрузку: *https://ftp.gnu.org/gnu/gmp/gmp-6.3.0.tar.xz*
Контрольная сумма MD5: `956dc04e864001a9c22429f761f2c283`

• Gperf (3.1) - 1,188 KB:
Домашняя страница: *https://www.gnu.org/software/gperf/*
Ссылка на загрузку: *https://ftp.gnu.org/gnu/gperf/gperf-3.1.tar.gz*
Контрольная сумма MD5: `9e251c0a618ad0824b51117d5d9db87e`

• Grep (3.11) - 1,664 KB:
Домашняя страница: *https://www.gnu.org/software/grep/*
Ссылка на загрузку: *https://ftp.gnu.org/gnu/grep/grep-3.11.tar.xz*
Контрольная сумма MD5: `7c9bbd74492131245f7cdb291fa142c0`

• Groff (1.23.0) - 7,259 KB:
Домашняя страница: *https://www.gnu.org/software/groff/*
Ссылка на загрузку: *https://ftp.gnu.org/gnu/groff/groff-1.23.0.tar.gz*
Контрольная сумма MD5: `5e4f40315a22bb8a158748e7d5094c7d`

• GRUB (2.12) - 6,524 KB:
Домашняя страница: *https://www.gnu.org/software/grub/*
Ссылка на загрузку: *https://ftp.gnu.org/gnu/grub/grub-2.12.tar.xz*
Контрольная сумма MD5: `60c564b1bdc39d8e43b3aab4bc0fb140`

• Gzip (1.13) - 819 KB:
Домашняя страница: *https://www.gnu.org/software/gzip/*
Ссылка на загрузку: *https://ftp.gnu.org/gnu/gzip/gzip-1.13.tar.xz*
Контрольная сумма MD5: `d5c9fc9441288817a4a0be2da0249e29`

• Iana-Etc (20240806) - 590 KB:
Домашняя страница: *https://www.iana.org/protocols*
Ссылка на загрузку: *https://github.com/Mic92/iana-etc/releases/download/20240806/iana-etc-20240806.tar.gz*
Контрольная сумма MD5: `ea3c37c00d22f1159fc3b7d988de8476`

• Inetutils (2.5) - 1,632 KB:
Домашняя страница: *https://www.gnu.org/software/inetutils/*
Ссылка на загрузку: *https://ftp.gnu.org/gnu/inetutils/inetutils-2.5.tar.xz*
Контрольная сумма MD5: `9e5a6dfd2d794dc056a770e8ad4a9263`

• Intltool (0.51.0) - 159 KB:
Домашняя страница: *https://freedesktop.org/wiki/Software/intltool*
Ссылка на загрузку: *https://launchpad.net/intltool/trunk/0.51.0/+download/intltool-0.51.0.tar.gz*
Контрольная сумма MD5: `12e517cac2b57a0121cda351570f1e63`

• IPRoute2 (6.10.0) - 900 KB:
Домашняя страница: *https://www.kernel.org/pub/linux/utils/net/iproute2/*
Ссылка на загрузку: *https://www.kernel.org/pub/linux/utils/net/iproute2/iproute2-6.10.0.tar.xz*
Контрольная сумма MD5: `6282e47de9c5b230e83537fba7181c9c`

• ISL (0.26) - 1,988 KB:
Домашняя страница: *http://libisl.sourceforge.io/*
Ссылка на загрузку: *https://libisl.sourceforge.io/isl-0.26.tar.xz*
Контрольная сумма MD5: `f2b56e8da688eba89838d414470da85f`

• Jinja2 (3.1.4) - 235 KB:
Домашняя страница: *https://jinja.palletsprojects.com/en/3.1.x/*
Ссылка на загрузку: *https://pypi.org/packages/source/J/Jinja2/jinja2-3.1.4.tar.gz*
Контрольная сумма MD5: `02ca9a6364c92e83d14b037bef4732bc`

• Kbd (2.6.4) - 1,470 KB:
Домашняя страница: *https://kbd-project.org/*
Ссылка на загрузку: *https://www.kernel.org/pub/linux/utils/kbd/kbd-2.6.4.tar.xz*
Контрольная сумма MD5: `e2fd7adccf6b1e98eb1ae8d5a1ce5762`

• Kmod (33) - 503 KB:
Домашняя страница: *https://github.com/kmod-project/kmod*
Ссылка на загрузку: *https://www.kernel.org/pub/linux/utils/kernel/kmod/kmod-33.tar.xz*
Контрольная сумма MD5: `c451c4aa61521adbe8af147f498046f8`

• Less (661) - 634 KB:
Домашняя страница: *https://www.greenwoodsoftware.com/less/*
Ссылка на загрузку: *https://www.greenwoodsoftware.com/less/less-661.tar.gz*
Контрольная сумма MD5: `44f54b6313c5d71fa1ac224d8d84766a`

• LFS-Bootscripts (20240825) - 34 KB:
Ссылка на загрузку: *https://mirror.linuxfromscratch.ru/lfs/downloads/12.2/lfs-bootscripts-20240825.tar.xz*
Контрольная сумма MD5: `d9c83d4dc68decf063f7da8c8db77f70`

• Libcap (2.70) - 187 KB:
Домашняя страница: *https://sites.google.com/site/fullycapable/*
Ссылка на загрузку: *https://www.kernel.org/pub/linux/libs/security/linux-privs/libcap2/libcap-2.70.tar.xz*
Контрольная сумма MD5: `df0e20c6eeca849347b87d5d6a8870c0`

• Libffi (3.4.6) - 1,360 KB:
Домашняя страница: *https://sourceware.org/libffi/*
Ссылка на загрузку: *https://github.com/libffi/libffi/releases/download/v3.4.6/libffi-3.4.6.tar.gz*
Контрольная сумма MD5: `b9cac6c5997dca2b3787a59ede34e0eb`

• Libpipeline (1.5.7) - 956 KB:
Домашняя страница: *https://libpipeline.nongnu.org/*
Ссылка на загрузку: *https://download.savannah.gnu.org/releases/libpipeline/libpipeline-1.5.7.tar.gz*
Контрольная сумма MD5: `1a48b5771b9f6c790fb4efdb1ac71342`

• Libtool (2.4.7) - 996 KB:
Домашняя страница: *https://www.gnu.org/software/libtool/*
Ссылка на загрузку: *https://ftp.gnu.org/gnu/libtool/libtool-2.4.7.tar.xz*
Контрольная сумма MD5: `2fc0b6ddcd66a89ed6e45db28fa44232`

• Libxcrypt (4.4.36) - 610 KB:
Домашняя страница: *https://github.com/besser82/libxcrypt/*
Ссылка на загрузку: *https://github.com/besser82/libxcrypt/releases/download/v4.4.36/libxcrypt-4.4.36.tar.xz*
Контрольная сумма MD5: `b84cd4104e08c975063ec6c4d0372446`

• Linux (6.10.5) - 141,739 KB:
Домашняя страница: *https://www.kernel.org/*
Ссылка на загрузку: *https://www.kernel.org/pub/linux/kernel/v6.x/linux-6.10.5.tar.xz*
Контрольная сумма MD5: `276ef1f11ed3713ec5d6f506ff55ac12`

Примечание

Ядро Linux обновляется достаточно часто из-за обнаружения уязвимостей в системе безопасности. Можно использовать последнюю стабильную версию ядра, если на странице с ошибками и рекомендациями по безопасности не указано иное.
Для пользователей, у которых ограниченный или тарифицируемый выход в интернет, и которые хотят обновить ядро Linux, можно скачать базовую версию ядра, а затем применить к ней патчи, которые могут быть загружены отдельно. Это может сэкономить немного времени или стоимость при обновлению до следующих версий.

• Lz4 (1.10.0) - 379 KB:
Домашняя страница: *https://lz4.org/*
Ссылка на загрузку: *https://github.com/lz4/lz4/releases/download/v1.10.0/lz4-1.10.0.tar.gz*
Контрольная сумма MD5: `dead9f5f1966d9ae56e1e32761e4e675`

• M4 (1.4.19) - 1,617 KB:
Домашняя страница: *https://www.gnu.org/software/m4/*
Ссылка на загрузку: *https://ftp.gnu.org/gnu/m4/m4-1.4.19.tar.xz*
Контрольная сумма MD5: `0d90823e1426f1da2fd872df0311298d`

• Make (4.4.1) - 2,300 KB:
Домашняя страница: *https://www.gnu.org/software/make/*
Ссылка на загрузку: *https://ftp.gnu.org/gnu/make/make-4.4.1.tar.gz*
Контрольная сумма MD5: `c8469a3713cbbe04d955d4ae4be23eeb`

• Man-DB (2.12.1) - 1,994 KB:
Домашняя страница: *https://www.nongnu.org/man-db/*
Ссылка на загрузку: *https://download.savannah.gnu.org/releases/man-db/man-db-2.12.1.tar.xz*
Контрольная сумма MD5: `7b044e5020aab89db41ac7ee59d6d84a`

• Man-pages (6.9.1) - 1,821 KB:
Домашняя страница: *https://www.kernel.org/doc/man-pages/*
Ссылка на загрузку: *https://www.kernel.org/pub/linux/docs/man-pages/man-pages-6.9.1.tar.xz*
Контрольная сумма MD5: `4d56775b6cce4edf1e496249e7c01c1a`

• MarkupSafe (2.1.5) - 19 KB:
Домашняя страница: *https://palletsprojects.com/p/markupsafe/*
Ссылка на загрузку: *https://pypi.org/packages/source/M/MarkupSafe/MarkupSafe-2.1.5.tar.gz*
Контрольная сумма MD5: `8fe7227653f2fb9b1ffe7f9f2058998a`

• Meson (1.5.1) - 2,205 KB:
Домашняя страница: *https://mesonbuild.com*
Ссылка на загрузку: *https://github.com/mesonbuild/meson/releases/download/1.5.1/meson-1.5.1.tar.gz*
Контрольная сумма MD5: `c4f2b3e5ea632685f61ba1b833c4905c`

• MPC (1.3.1) - 756 KB:
Домашняя страница: *https://www.multiprecision.org/*
Ссылка на загрузку: *https://ftp.gnu.org/gnu/mpc/mpc-1.3.1.tar.gz*
Контрольная сумма MD5: `5c9bc658c9fd0f940e8e3e0f09530c62`

• MPFR (4.2.1) - 1,459 KB:
Домашняя страница: *https://www.mpfr.org/*
Ссылка на загрузку: *https://ftp.gnu.org/gnu/mpfr/mpfr-4.2.1.tar.xz*
Контрольная сумма MD5: `523c50c6318dde6f9dc523bc0244690a`

• Ncurses (6.5) - 2,156 KB:
Домашняя страница: *https://www.gnu.org/software/ncurses/*
Ссылка на загрузку: *https://invisible-mirror.net/archives/ncurses/ncurses-6.5.tar.gz*
Контрольная сумма MD5: `ac2d2629296f04c8537ca706b6977687`

• Ninja (1.12.1) - 235 KB:
Домашняя страница: *https://ninja-build.org/*
Ссылка на загрузку: *https://github.com/ninja-build/ninja/archive/v1.12.1/ninja-1.12.1.tar.gz*
Контрольная сумма MD5: `6288992b05e593a391599692e2f7e490`

• OpenSSL (3.3.1) - 17,633 KB:
Домашняя страница: *https://www.openssl.org/*
Ссылка на загрузку: *https://www.openssl.org/source/openssl-3.3.1.tar.gz*
Контрольная сумма MD5: `8a4342b399c18f870ca6186299195984`

• Patch (2.7.6) - 766 KB:
Домашняя страница: *https://savannah.gnu.org/projects/patch/*
Ссылка на загрузку: *https://ftp.gnu.org/gnu/patch/patch-2.7.6.tar.xz*
Контрольная сумма MD5: `78ad9937e4caadcba1526ef1853730d5`

• Perl (5.40.0) - 13,481 KB:
Домашняя страница: *https://www.perl.org/*
Ссылка на загрузку: *https://www.cpan.org/src/5.0/perl-5.40.0.tar.xz*
Контрольная сумма MD5: `cfe14ef0709b9687f9c514042e8e1e82`

• Pkgconf (2.3.0) - 309 KB:
Домашняя страница: *https://github.com/pkgconf/pkgconf*
Ссылка на загрузку: *https://distfiles.ariadne.space/pkgconf/pkgconf-2.3.0.tar.xz*
Контрольная сумма MD5: `833363e77b5bed0131c7bc4cc6f7747b`

• Procps (4.0.4) - 1,369 KB:
Домашняя страница: *https://gitlab.com/procps-ng/procps/*
Ссылка на загрузку: *https://sourceforge.net/projects/procps-ng/files/Production/procps-ng-4.0.4.tar.xz*
Контрольная сумма MD5: `2f747fc7df8ccf402d03e375c565cf96`

• Psmisc (23.7) - 423 KB:
Домашняя страница: *https://gitlab.com/psmisc/psmisc*
Ссылка на загрузку: *https://sourceforge.net/projects/psmisc/files/psmisc/psmisc-23.7.tar.xz*
Контрольная сумма MD5: `53eae841735189a896d614cba440eb10`

• Python (3.12.5) - 19,944 KB:
Домашняя страница: *https://www.python.org/*
Ссылка на загрузку: *https://www.python.org/ftp/python/3.12.5/Python-3.12.5.tar.xz*
Контрольная сумма MD5: `02c7d269e077f4034963bba6befdc715`

• Python Documentation (3.12.5) - 8,188 KB:
Ссылка на загрузку: *https://www.python.org/ftp/python/doc/3.12.5/python-3.12.5-docs-html.tar.bz2*
Контрольная сумма MD5: `52274d813236ca4a972fb6988480dc56`

• Readline (8.2.13) - 2,974 KB:
Домашняя страница: *https://tiswww.case.edu/php/chet/readline/rltop.html*
Ссылка на загрузку: *https://ftp.gnu.org/gnu/readline/readline-8.2.13.tar.gz*
Контрольная сумма MD5: `05080bf3801e6874bb115cd6700b708f`

• Sed (4.9) - 1,365 KB:
Домашняя страница: *https://www.gnu.org/software/sed/*
Ссылка на загрузку: *https://ftp.gnu.org/gnu/sed/sed-4.9.tar.xz*
Контрольная сумма MD5: `6aac9b2dbafcd5b7a67a8a9bcb8036c3`

• Setuptools (72.2.0) - 2,363 KB:
Домашняя страница: *https://pypi.org/project/setuptools/*
Ссылка на загрузку: *https://pypi.org/packages/source/s/setuptools/setuptools-72.2.0.tar.gz*
Контрольная сумма MD5: `2e0ffd0f6fc632a11442b79d9b1c68bd`

• Shadow (4.16.0) - 2,154 KB:
Домашняя страница: *https://github.com/shadow-maint/shadow/*
Ссылка на загрузку: *https://github.com/shadow-maint/shadow/releases/download/4.16.0/shadow-4.16.0.tar.xz*
Контрольная сумма MD5: `eb70bad3316d08f0d3bb3d4bbeccb3b4`

• Sysklogd (2.6.1) - 452 KB:
Домашняя страница: *https://www.infodrom.org/projects/sysklogd/*
Ссылка на загрузку: *https://github.com/troglobit/sysklogd/releases/download/v2.6.1/sysklogd-2.6.1.tar.gz*
Контрольная сумма MD5: `dcf0836a0fcc6568efaad230850d9c86`

• Systemd (256.4) - 15,291 KB:
Домашняя страница: *https://www.freedesktop.org/wiki/Software/systemd/*
Ссылка на загрузку: *https://github.com/systemd/systemd/archive/v256.4/systemd-256.4.tar.gz*
Контрольная сумма MD5: `03bd1ff158ec0bc55428c77a8f8495bd`

• Systemd Man Pages (256.4) - 676 KB:
Домашняя страница: *https://www.freedesktop.org/wiki/Software/systemd/*
Ссылка на загрузку: *https://anduin.linuxfromscratch.org/LFS/systemd-man-pages-256.4.tar.xz*
Контрольная сумма MD5: `8dbcf0ff0d8e5e9d3565f9d2fc153310`

Примечание
Команда Linux From Scratch генерирует собственный архив справочных страниц, используя исходный код systemd. Это делается для того, чтобы избежать ненужных зависимостей.

• SysVinit (3.10) - 235 KB:
Домашняя страница: *https://savannah.nongnu.org/projects/sysvinit*
Ссылка на загрузку: *https://github.com/slicer69/sysvinit/releases/download/3.10/sysvinit-3.10.tar.xz*
Контрольная сумма MD5: `b8fbe11062cf16d3b6a3709b7f6978d2`

• Tar (1.35) - 2,263 KB:
Домашняя страница: *https://www.gnu.org/software/tar/*
Ссылка на загрузку: *https://ftp.gnu.org/gnu/tar/tar-1.35.tar.xz*
Контрольная сумма MD5: `a2d8042658cfd8ea939e6d911eaf4152`

• Tcl (8.6.14) - 11,355 KB:
Домашняя страница: *https://tcl.sourceforge.net/*
Ссылка на загрузку: *https://downloads.sourceforge.net/tcl/tcl8.6.14-src.tar.gz*
Контрольная сумма MD5: `c30b57c6051be28fa928d09aca82841e`

• Tcl Documentation (8.6.14) - 1,167 KB:
Ссылка на загрузку: *https://downloads.sourceforge.net/tcl/tcl8.6.14-html.tar.gz*
Контрольная сумма MD5: `5467198f8d57c54835bf80b98ffb0170`

• Texinfo (7.1) - 5,416 KB:
Домашняя страница: *https://www.gnu.org/software/texinfo/*
Ссылка на загрузку: *https://ftp.gnu.org/gnu/texinfo/texinfo-7.1.tar.xz*
Контрольная сумма MD5: `edd9928b4a3f82674bcc3551616eef3b`

• Time Zone Data (2024a) - 444 KB:
Домашняя страница: *https://www.iana.org/time-zones*
Ссылка на загрузку: *https://www.iana.org/time-zones/repository/releases/tzdata2024a.tar.gz*
Контрольная сумма MD5: `2349edd8335245525cc082f2755d5bf4`

• Udev-lfs Tarball (udev-lfs-20230818) - 10 KB:
Ссылка на загрузку: *https://anduin.linuxfromscratch.org/LFS/udev-lfs-20230818.tar.xz*
Контрольная сумма MD5: `acd4360d8a5c3ef320b9db88d275dae6`

• Util-linux (2.40.2) - 8,648 KB:
Домашняя страница: *https://git.kernel.org/pub/scm/utils/util-linux/util-linux.git/*
Ссылка на загрузку: *https://www.kernel.org/pub/linux/utils/util-linux/v2.40/util-linux-2.40.2.tar.xz*
Контрольная сумма MD5: `88faefc8fefced097e58142077a3d14e`

• Vim (9.1.0660) - 17,629 KB:
Домашняя страница: *https://www.vim.org*
Ссылка на загрузку: *https://github.com/vim/vim/archive/v9.1.0660/vim-9.1.0660.tar.gz*
Контрольная сумма MD5: `c512a99b3704f193be1a181cc644b2b2`

Примечание
Версия vim меняется ежедневно. Чтобы получить последнюю версию, перейдите на *https://github.com/vim/vim/tags*.

• Wheel (0.44.0) - 99 KB:
Домашняя страница: *https://pypi.org/project/wheel/*
Ссылка на загрузку: *https://pypi.org/packages/source/w/wheel/wheel-0.44.0.tar.gz*
Контрольная сумма MD5: `440ff4fe51579b7ed16f02af8f8d9494`

• XML::Parser (2.47) - 276 KB:
Домашняя страница: *https://github.com/chorny/XML-Parser*
Ссылка на загрузку: *https://cpan.metacpan.org/authors/id/T/TO/TODDR/XML-Parser-2.47.tar.gz*
Контрольная сумма MD5: `89a8e82cfd2ad948b349c0a69c494463`

• Xz Utils (5.6.2) - 1,277 KB:
Домашняя страница: *https://tukaani.org/xz*
Ссылка на загрузку: *https://github.com//tukaani-project/xz/releases/download/v5.6.2/xz-5.6.2.tar.xz*
Контрольная сумма MD5: `bbf73fb28425cebb854328599f85c4cf`

• Zlib (1.3.1) - 1,478 KB:
Домашняя страница: *https://zlib.net/*
Ссылка на загрузку: *https://zlib.net/fossils/zlib-1.3.1.tar.gz*
Контрольная сумма MD5: `9855b6d802d7fe5b7bd5b196a2271655`

• Zstd (1.5.6) - 2,351 KB:
Домашняя страница: *https://facebook.github.io/zstd/*
Ссылка на загрузку: *https://github.com/facebook/zstd/releases/download/v1.5.6/zstd-1.5.6.tar.gz*
Контрольная сумма MD5: `5a473726b3445d0e5d6296afd1ab6854`

Общий размер пакетов: примерно 519 MB

3.3. Необходимые патчи

В дополнение к пакетам требуется несколько патчей. Эти патчи исправляют ошибки в пакетах, которые должны быть исправлены сопровождающими. Патчи также вносят небольшие изменения, облегчающие работу с пакетами. Для сборки системы LFS потребуются следующие патчи:

• Bzip2 Documentation Patch - 1.6 KB:
Ссылка на загрузку: *https://mirror.linuxfromscratch.ru/patches/lfs/12.2/bzip2-1.0.8-install_docs-1.patch*
Контрольная сумма MD5: `6a5ac7e89b791aae556de0f745916f7f`

• Coreutils Internationalization Fixes Patch - 164 KB:
Ссылка на загрузку: *https://mirror.linuxfromscratch.ru/patches/lfs/12.2/coreutils-9.5-i18n-2.patch*
Контрольная сумма MD5: `58961caf5bbdb02462591fa506c73b6d`

• Expect GCC14 Patch - 7.8 KB:
Download: *https://mirror.linuxfromscratch.ru/patches/lfs/12.2/expect-5.45.4-gcc14-1.patch*
MD5 sum: `0b8b5ac411d011263ad40b0664c669f0`

• Glibc FHS Patch - 2.8 KB:
Ссылка на загрузку: *https://mirror.linuxfromscratch.ru/patches/lfs/12.2/glibc-2.40-fhs-1.patch*
Контрольная сумма MD5: `9a5997c3452909b1769918c759eff8a2`

• Kbd Backspace/Delete Fix Patch - 12 KB:
Ссылка на загрузку: *https://mirror.linuxfromscratch.ru/patches/lfs/12.2/kbd-2.6.4-backspace-1.patch*
Контрольная сумма MD5: `f75cca16a38da6caa7d52151f7136895`

• SysVinit Consolidated Patch - 2.5 KB:
Ссылка на загрузку: *https://mirror.linuxfromscratch.ru/patches/lfs/12.2/sysvinit-3.10-consolidated-1.patch*
Контрольная сумма MD5: `17ffccbb8e18c39e8cedc32046f3a475`

Общий размер этих патчей: примерно 190.7 KB

Помимо указанных выше обязательных исправлений, существует ряд необязательных патчей, созданных сообществом LFS. Эти необязательные исправления решают незначительные проблемы или включают функции, которые не включены по умолчанию. Не стесняйтесь просматривать базу данных исправлений, расположенную по адресу *https://mirror.linuxfromscratch.ru/patches/downloads/*, и применять патчи, необходимые вашей системе.

Глава 4. Заключительный этап подготовки

4.1. Введение

В этой главе мы выполним несколько дополнительных настроек для подготовки к сборке временной системы. Мы создадим несколько каталогов в `$LFS` (в котором установим временные инструменты), добавим непривилегированного пользователя и настроим окружение для этого пользователя. Кроме этого, будут даны пояснения по стандартной единице времени сборки, или «SBU», которую мы используем для измерения времени необходимого для сборки пакетов LFS, и предоставим некоторую информацию о наборах тестов.

4.2. Создание ограниченной иерархии папок в файловой системе LFS

В этом разделе мы начинаем заполнять файловую систему LFS элементами, которые будут основой конечной системы Linux. Первым шагом является создание ограниченной иерархии каталогов, чтобы программы, скомпилированные в Глава 6 (а также glibc и libstdc++ в Глава 5), могли быть установлены в их конечном расположении. Это необходимо для того, чтобы эти временные программы были перезаписаны при сборке окончательных версий в Глава 8.

Создайте необходимую иерархию каталогов, выполнив следующую команду от имени `root`:

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

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

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

```
mkdir -pv $LFS/usr/lib32
ln -sv usr/lib32 $LFS/lib32
```

Программы в Глава 6 будут скомпилированы с помощью кросс-компилятора (более подробная информация приведена в разделе Технические примечания по сборочным инструментам). Чтобы отделить кросс-компилятор от других программ, он будет установлен в специальный каталог. Создайте этот каталог с помощью следующей команды:

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

Примечание

Редакторы LFS намеренно решили не использовать каталог `/usr/lib64` . В процессе сборки предпринимается ряд шагов, чтобы убедиться, что набор инструментов не будет его использовать. Если по какой-либо причине этот каталог появится (это может произойти, если вы допустили ошибку, следуя инструкциям, или потому что вы установили бинарный пакет, создавший его после сборки LFS), это может привести к поломке вашей системы. Вы должны быть уверены, что этого каталога не существует.

4.3. Создание пользователя LFS

При входе в систему под учетной записью `root` допущение одной ошибки может привести к повреждению или разрушению системы. Поэтому пакеты в следующих двух главах собираются из-под учетной записи непривилегированного пользователя. Вы можете использовать свое собственное имя пользователя, но чтобы

упростить настройку рабочей среды, создайте нового пользователя с именем `lfs`, который является членом одноименной группы и выполняйте команды из-под этой учетной записи в процессе установки. От имени пользователя `root` выполните следующие команды, чтобы добавить нового пользователя:

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

Значение параметров командной строки:

`-s /bin/bash`
Устанавливает **bash** оболочкой по умолчанию для пользователя `lfs`.

`-g lfs`
Эта опция добавляет пользователя `lfs` в группу `lfs`.

`-m`
Создает домашний каталог для пользователя `lfs`.

`-k /dev/null`
Этот параметр предотвращает возможное копирование файлов из предустановленного набора каталогов (по умолчанию `/etc/skel`) путем изменения местоположения ввода на специальное null-устройство.

`lfs`
Это имя нового пользователя.

Если вы хотите войти в систему как `lfs` или переключиться на `lfs` из учетной записи непривилегированного пользователя (в отличие от переключения на пользователя `lfs` при входе в систему как `root`, для которого не требуется пароль пользователя `lfs`), вам необходимо установить пароль для `lfs`. Выполните следующую команду от имени пользователя `root`, чтобы установить пароль:

```
passwd lfs
```

Предоставьте пользователю `lfs` полный доступ ко всем каталогам в папке `$LFS`, назначив `lfs` владельцем:

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

```
chown -v lfs $LFS/lib32
```

Примечание

В некоторых хост-системах следующая команда не выполняется должным образом и приостанавливает вход пользователя `lfs` в фоновом режиме. Если подсказка "lfs:~$" не появляется сразу, ввод команды **fg** устранит проблему.

Затем запустите оболочку, работающую от имени пользователя `lfs`. Это можно сделать, войдя в систему как `lfs` на виртуальной консоли или с помощью следующей команды замены/переключения пользователя:

```
su - lfs
```

Аргумент «-» передает значение команде **su** для запуска оболочки входа в систему, а не обычной оболочки. Разница между этими двумя типами оболочек подробно описана в *bash(1)* и **info bash**.

4.4. Настройка окружения

Настроим хорошо работающее окружение, создав два новых файла запуска для оболочки **bash**. Войдя в систему как пользователь `lfs`, введите следующую команду, чтобы создать новый `.bash_profile` :

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

При входе в систему под учетной записью пользователя `lfs` или при переключении на `lfs`, используя команду **su** с опцией «-», начальная оболочка представляет собой оболочку *login*, которая читает данные из `/etc/profile` хоста (который, вероятно, содержит некоторые настройки и переменные среды), а затем `.bash_profile`. Команда **exec env -i.../bin/bash** в файле `.bash_profile` заменяет запущенную оболочку новой, не содержащей переменные среды, за исключением переменных `HOME`, `TERM`, и `PS1`. Это гарантирует, что никакие нежелательные и потенциально опасные переменные среды из хост-системы не попадут в среду сборки.

Новый экземпляр оболочки представляет собой *non-login* оболочку, которая не считывает и не выполняет содержимое файлов `/etc/profile` и `.bash_profile`, а вместо этого выполняет чтение из файла `.bashrc`. Создайте файл `.bashrc`:

```
cat > ~/.bashrc << "EOF"
set +h
umask 022
LFS=/mnt/lfs
LC_ALL=POSIX
LFS_TGT=x86_64-lfs-linux-gnu
LFS_TGT32=i686-lfs-linux-gnu
LFS_TGTX32=x86_64-lfs-linux-gnux32
PATH=/usr/bin
if [ ! -L /bin ]; then PATH=/bin:$PATH; fi
PATH=$LFS/tools/bin:$PATH
CONFIG_SITE=$LFS/usr/share/config.site
export LFS LC_ALL LFS_TGT LFS_TGT32 LFS_TGTX32 PATH
EOF
```

Значение настроек в `.bashrc`

`set +h`

Команда **set +h** отключает хэш-функцию **bash**. Хеширование является полезной функцией — **bash** использует хеш-таблицу для запоминания полного пути к исполняемому файлу, чтобы избежать многократного поиска одного и того же исполняемого файла в переменной окружения `PATH`. Однако новые инструменты требуется использовать сразу же после их установки. Отключение хэш-функции, заставляет оболочку искать переменную окружения `PATH`, всякий раз, когда программу необходимо запустить. Таким образом, оболочка найдет вновь скомпилированные инструменты в `$LFS/tools/bin`, как только они станут доступны, не запоминая предыдущую версию той же программы, предоставленную хост-дистрибутивом, в `/usr/bin` или `/bin`.

`umask 022`

Установка значения пользовательской маски создания файлов (umask) 022 гарантирует, что вновь созданные файлы и каталоги доступны для записи только их владельцу, но будут доступны для чтения и выполнения остальным пользователям (при условии, что системный вызов *open(2)* использует режимы по умолчанию, новые файлы получат разрешения 644, а каталоги 755).

`LFS=/mnt/lfs`

Переменная окружения `LFS` должна указывать на выбранную точку монтирования.

`LC_ALL=POSIX`

Переменная `LC_ALL` управляет локализацией определенных программ, и формирует сообщения в соответствии с локализацией указанной страны. Установка в `LC_ALL` значения «POSIX» или «C» (они эквивалентны) гарантирует, что все будет работать должным образом в среде кросс-компиляции.

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

Переменная `LFS_TGT` устанавливает нестандартное, но совместимое описание компьютера для использования при создании кросс-компилятора и компоновщика, а также при кросс-компиляции временного набора инструментов. Дополнительная информация об этом представлена в Технические примечания по сборочным инструментам.

PATH=/usr/bin

Многие современные дистрибутивы Linux объединили `/bin` и `/usr/bin` . В этом случае стандартной переменной `PATH` необходимо установить значение `/usr/bin/` для окружения из Глава 6. Когда это не так, следующая строка добавит `/bin` к пути.

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

Если `/bin` не является символической ссылкой, то его необходимо добавить в переменную `PATH`.

PATH=$LFS/tools/bin:$PATH

Поместив `$LFS/tools/bin` перед стандартным `PATH`, кросс-компилятор, установленный в начале Глава 5, будет обнаружен оболочкой сразу после его установки. Это, в сочетании с отключением хеширования, ограничивает риск использования компилятора хоста вместо кросс-компилятора.

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

В Глава 5 и Глава 6, если эта переменная не задана, сценарии **configure** могут попытаться загрузить элементы конфигурации, специфичные для некоторых дистрибутивов, из `/usr/share/config.site` в хост-системе. Переопределите её, чтобы предотвратить потенциальное влияние хоста.

export ...

Приведенные выше команды установили некоторые переменные, чтобы сделать их видимыми в любых вложенных оболочках, мы экспортируем их.

Важно

Некоторые коммерческие дистрибутивы добавляют недокументированный экземпляр `/etc/bash.bashrc` для инициализации **bash**. Этот файл потенциально может изменить среду пользователя `lfs` таким образом, что это может повлиять на сборку важных пакетов LFS. Чтобы убедиться, что пользовательская среда `lfs` чиста, проверьте наличие файла `/etc/bash.bashrc` и, если он есть, переименуйте его. От имени пользователя `root`, запустите:

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

Когда пользователь `lfs` больше не нужен (в начале Глава 7) вы можете безопасно восстановить `/etc/bash.bashrc` (по желанию).

Обратите внимание, что пакет LFS Bash, который мы создадим в Раздел 8.37, «Bash-5.2.32», не настроен на загрузку или выполнение `/etc/bash.bashrc` , поэтому этот файл бесполезен в готовой системе LFS.

Для многих современных систем с несколькими процессорами (или ядрами) время компиляции пакета можно сократить, выполнив «параллельную сборку», либо установив переменную среды, либо сообщив программе **make**, сколько ядер задействовать для сборки. Например, процессор Intel Core i9-13900K имеет 8 ядер P (производительность) и 16 ядер E (энергоэффективность), ядро P может одновременно запускать два потока, поэтому каждое ядро P моделируется ядром Linux как два логических ядра. В результате получается 32 логических ядра. Очевидный способ задействовать все эти логические ядра - разрешить **make** создавать до 32 заданий сборки. Это можно сделать, передав параметр *-j32* команде **make**:

```
make -j32
```

Или установите переменную окружения `MAKEFLAGS`, и ее содержимое будет автоматически использоваться **make** в качестве параметров командной строки:

```
export MAKEFLAGS=-j32
```

Важно

Никогда не передавайте параметр `-j` без номера в **make** и не устанавливайте такой параметр в `MAKEFLAGS`. Иначе **make** будет создавать бесконечные задания сборки, что вызовет проблемы со стабильностью системы.

Чтобы использовать все логические ядра, доступные для сборки пакетов в Глава 5 и Глава 6, укажите параметр `MAKEFLAGS` в `.bashrc сейчас` :

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

Замените `$(nproc)` количеством логических ядер, которые вы хотите использовать, если вы планируете использовать не все логические ядра.

Наконец, чтобы убедиться, что среда полностью подготовлена для сборки временных инструментов, перечитайте только что созданный профиль пользователя:

```
source ~/.bash_profile
```

4.5. О SBU (Стандартная единица времени сборки)

Многие люди хотели бы знать заранее, сколько примерно времени потребуется для компиляции и установки каждого пакета. Поскольку Linux From Scratch может быть собран на различных системах, невозможно дать точную оценку времени. Сборка самого большого пакета (gcc) займет около 5 минут на быстрых системах, но может занять несколько дней на более медленных компьютерах! Вместо фактического времени в книге используется показатель "стандартная единица времени сборки" (SBU).

Показатель SBU рассчитывается следующим образом. Первым пакетом, который нужно скомпилировать, является binutils в Глава 5. Время, необходимое для компиляции этого пакета с использованием одного ядра, будет называться стандартной единицей времени сборки или SBU. Время компиляции остальных пакетов будет рассчитано относительно этого времени.

Например, рассмотрим пакет, время компиляции которого составляет 4,5 SBU. Это означает, что если вашей системе потребовалось 4 минуты для компиляции и сборки первого прохода binutils, то для сборки этого пакета потребуется *примерно* 18 минут. К счастью, в большинстве случаев, время сборки меньше, чем у binutils.

В целом, величина SBU не совсем точна, поскольку она зависит от множества факторов, начиная от версии GCC хост-системы. Она приведены здесь, чтобы дать оценку того, сколько времени может потребоваться для сборки пакета, но в некоторых случаях цифры могут отличаться на десятки минут.

В современных материнских платах есть возможность управлять тактовой частотой процессора. Это можно сделать с помощью команды, такой как, **powerprofilesctl**. Данная возможность отсутствует в LFS, но может быть доступна в хост-дистрибутиве. После сборки LFS управление производительностью и электропитанием можно добавить в систему с помощью инструкций, описанных на странице *BLFS power-profiles-daemon*.

Перед измерением времени сборки любого пакета рекомендуется задействовать профиль энергопотребления системы, настроенный на максимальную производительность (и максимальное энергопотребление).

В противном случае измеренное значение SBU может быть неточным, поскольку система может по-разному реагировать на сборку binutils-pass1 или других пакетов.

Имейте в виду, что даже если для обоих пакетов используется один и тот же профиль, все равно могут возникнуть существенные неточности, поскольку система может реагировать медленнее, если система простаивает при запуске процедуры сборки. Установка для профиля питания режима «производительность» сведет к минимуму эту проблему. И, очевидно, это также ускорит сборку LFS.

Если **powerprofilesctl** доступен, введите команду **powerprofilesctl set performance**, чтобы выбрать профиль производительность. Некоторые дистрибутивы для управления профилями предоставляют команду **tuned-adm** вместо **powerprofilesctl**, в этих дистрибутивах введите команду **tuned-adm profile throughput-performance**, чтобы выбрать профиль производительность.

Примечание

Когда используется несколько ядер, единицы измерения SBU будут различаться еще больше, чем обычно. В некоторых случаях make просто завершится ошибкой. Анализ выходных данных процесса сборки также будет более сложным, поскольку строки разных потоков будут чередоваться. Если вы столкнулись с проблемой на этапе сборки, вернитесь к сборке на одном ядре, чтобы проанализировать сообщения об ошибках.

Представленные здесь значения времени для всех пакетов (за исключением binutils-pass1, который собирается на одном ядре) рассчитаны при использовании четырех ядер (-j4). Время, указанное в главе 8, также включает время выполнения регрессионных тестов для пакета, если не указано иное.

4.6. О наборах тестов

Большинство пакетов предоставляют набор тестов. Запуск набора тестов для только что собранного пакета — хорошая идея, потому что он может обеспечить «проверку работоспособности», указывающую, что все скомпилировано правильно. Набор тестов, который проходит свой набор проверок, обычно доказывает, что пакет работает так, как задумал разработчик. Однако это не гарантирует, что пакет полностью без ошибок.

Некоторые наборы тестов более важны, чем другие. Например, наборы тестов для основных инструментов — GCC, binutils и glibc — имеют первостепенное значение из-за их центральной роли в правильно функционирующей системе. Выполнение наборов тестов для GCC и glibc может занять очень много времени, особенно на медленном оборудовании, но их выполнение настоятельно рекомендуется.

Примечание

Запуск наборов тестов, описанных в Глава 5 и Глава 6, не имеет смысла, поскольку программы компилируются с помощью кросс-компилятора, они, вероятно, не могут работать на хосте сборки.

Распространенной проблемой при запуске наборов тестов для binutils и GCC является нехватка псевдотерминалов (PTY). Это может привести к большому количеству неудачных тестов. Причин может быть несколько, но наиболее вероятная причина заключается в том, что в хост-системе неправильно настроена файловая система devpts. Этот вопрос более подробно обсуждается на странице *https://mirror.linuxfromscratch.ru/lfs/faq.html#no-ptys*.

Иногда наборы тестов не работают, по причинам, о которых знают разработчики и которые они считают некритичными. Просмотрите журналы, расположенные по адресу *https://mirror.linuxfromscratch.ru/lfs/build-logs/12.2/*, чтобы проверить, ожидаются ли сбои. Этот сайт актуален для всех наборов тестов, описанных в книге.

Часть III. Сборка кросс-компилятора и набора временных инструментов

Важный предварительный материал

Введение

Эта часть разделена на три этапа: во-первых, сборка кросс-компилятора и связанных с ним библиотек; во-вторых, использование этого набора инструментов для сборки нескольких утилит таким образом, чтобы изолировать их от основного дистрибутива; в-третьих, вход в среду chroot (что ещё больше улучшает изоляцию от хоста), и сборка оставшихся инструментов, необходимых для создания конечной системы.

Важно

Именно здесь начинается настоящая работа по сборке новой системы. Требуется очень тщательно следить за тем, чтобы инструкции выполнялись точно так, как они приведены в книге. Вы должны попытаться понять, что они делают, и каким бы ни было ваше желание скорее закончить сборку, вам следует воздержаться от слепого набора команд. Читайте документацию, если вы что-то не понимаете. Кроме того, следите за результатом выполнения команд, отправляя лог в файл с помощью утилиты **tee**. Это упрощает отладку, если что-то пойдет не так.

Следующий раздел представляет собой техническое введение в процесс сборки, а следующий за ним, содержит **очень важные** общие инструкции по компиляции.

Технические примечания по сборочным инструментам

В этом разделе объясняются причины и некоторые технические детали, лежащие в основе сборки пакетов. Не обязательно сразу понимать все, что содержится в этом разделе. Большая часть этой информации станет более понятной после выполнения фактической сборки. Возвращайтесь и перечитывайте этот раздел в любое время по ходу сборки.

Основная задача Глава 5 и Глава 6 состоит в том, чтобы создать временную область, содержащую заведомо исправный набор инструментов, которые можно изолировать от хост-системы. Использовании команды **chroot** в последующих главах, обеспечит чистую и безотказную сборку целевой системы LFS. Процесс сборки разработан таким образом, чтобы свести к минимуму риски для новых читателей и в то же время обеспечить наибольшую образовательную ценность.

Сборка инструментария основана на процессе *кросс-компиляции*. Кросс-компиляция обычно используется для сборки компилятора и его инструментов для машины, отличной от той, которая используется для сборки. Строго говоря, это не требуется для LFS, так как машина, на которой будет работать новая система, та же, что и используемая для сборки. Но у кросс-компиляции есть большое преимущество, заключающееся в том, что все, что подвергается кросс-компиляции, не будет зависеть от окружения хоста.

О кросс-компиляции

Примечание

Книга LFS не является руководством и не содержит общего руководства по созданию кросс (или собственного) тулчейна. Не используйте команды из книги для кросс-тулчейна, который планируете использовать для каких-либо других целей, кроме создания LFS, если у вас нет полного понимания, что вы делаете.

Кросс-компиляция включает в себя некоторые концепции, которые сами по себе заслуживают отдельного раздела. Хотя этот раздел можно пропустить при первом чтении, возвращение к нему позже будет полезно для полного понимания процесса.

Давайте определим некоторые термины, используемые в этом контексте.

сборщик
: это машина, на которой мы собираем программы. Обратите внимание, что этот компьютер упоминается как «хост» в других разделах.

хост
: это машина/система, на которой будут выполняться встроенные программы. Обратите внимание, что используемое здесь значение слова «хост» отличается от того, которое применяется в других разделах.

цель
: используется только для компиляторов. Это машина, для которой компилятор создает код. Он может отличаться как от «сборщика», так и от «хоста».

В качестве примера представим следующий сценарий (иногда называемый «канадским крестом»): у нас есть компилятор на медленной машине, назовем ее машиной A и компилятор ccA. У нас также есть быстрая машина (B), но без компилятора, и мы хотим создать код для другой медленной машины (C). Чтобы собрать компилятор для машины C, у нас будет три этапа:

Этап	Сборщик	Хост	Цель	Действие
1	A	A	B	Сборка кросс-компилятора cc1 с использованием ccA на машине A
2	A	B	C	Сборка кросс-компилятора cc2 с использованием cc1 на машине A
3	B	C	C	Сборка компилятора ccC с использованием cc2 на машине B

Затем все другие программы, необходимые для машины C, могут быть скомпилированы с помощью cc2 на быстрой машине B. Обратите внимание, что до тех пор, пока B не может запускать программы, собранные для C, нет способа протестировать программы, пока не будет запущена сама машина C. Например, чтобы запустить набор тестов на ccC мы можем добавить четвертый этап:

Этап	Сборщик	Хост	Цель	Действие
4	C	C	C	Пересобрать и протестировать ccC, используя ccC на машине C

В приведенном выше примере только cc1 и cc2 являются кросс-компиляторами, то есть они создают код для машины, отличной от той, на которой они выполняются. Компиляторы ccA и ccC создают код для машины, на которой они выполняются. Такие компиляторы называются *нативными* компиляторами.

Реализация кросс-компиляции для LFS

Примечание

Все кросс-компилируемые пакеты в этой книге используют систему сборки на основе autoconf. Система сборки на основе autoconf принимает типы систем вида cpu-vendor-kernel-os, называемые системным триплетом. Поскольку поле vendor часто не содержит значения, autoconf позволяет вам опустить его.

Проницательный читатель может задаться вопросом, почему название «триплет» применяется к имени из четырех компонентов. Поле kernel и поле os ранее применялись как единый элемент: «system». Такая форма с тремя полями все еще актуальна для некоторых систем, например, `x86_64-unknown-freebsd` . Но две системы могут использовать одно и то же ядро и все же быть слишком разными, чтобы использовать одинаковый триплет для их описания. Например, Android, работающий на мобильном телефоне полностью отличается от Ubuntu, работающей на ARM64 сервере, хотя они оба работают на одном и том же типе процессора (ARM64) и с одним ядром (Linux).

Без слоя эмуляции вы не сможете запустить исполняемый файл с сервера на мобильном телефоне и наоборот. Итак, поле «system» было разделено на поля kernel и os, чтобы однозначно их интерпретировать. В нашем примере Android обозначается как `aarch64-unknown-linux-android` , а Ubuntu `aarch64-unknown-linux-gnu` .

Слово «триплет» сохранилось в лексиконе. Простой способ определить триплет вашей машины — запустить скрипт **config.guess**, который входит в исходный код многих пакетов. Распакуйте исходники binutils и запустите скрипт: **`./config.guess`**, обратите внимание на вывод. Например, для 32-разрядного процессора Intel вывод будет *i686-pc-linux-gnu*. В 64-битной системе это будет *x86_64-pc-linux-gnu*. В большинстве систем Linux используют еще более простую команду **gcc -dumpmachine**, которая предоставит вам аналогичную информацию.

Вы также должны знать имя динамического компоновщика платформы, часто называемого динамическим загрузчиком (не путать со стандартным компоновщиком **ld**, который является частью binutils). Динамический компоновщик, предоставляемый glibc, находит и загружает общие библиотеки, необходимые программе, подготавливает программу к запуску, а затем запускает ее. Имя динамического компоновщика для 32-разрядной машины Intel — `ld-linux.so.2` , а для 64-разрядных систем — `ld-linux-x86-64.so.2` . Надежный способ определить имя динамического компоновщика — проверить случайный двоичный файл из хост-системы, выполнив следующую команду: **`readelf -l <имя исполняемого файла> | grep interpreter`** и зафиксировать результат. Официальный источник, охватывающий все платформы, находится на *вики-странице Glibc*.

Чтобы сымитировать кросс-компиляцию в LFS, имя триплета хоста немного подкорректировали, изменив поле "vendor" в переменной `LFS_TGT` таким образом, чтобы оно указывало "lfs". Мы также используем параметр `--with-sysroot` при сборке кросс-компоновщика и кросс-компилятора, чтобы сообщить им, где найти необходимые файлы хоста. Это гарантирует, что ни одна из программ, входящих в Глава 6, не сможет ссылаться на библиотеки на машине сборки. Для корректной работы, обязательны всего два этапа, еще один рекомендуется для тестирования:

Этап	Сборщик	Хост	Цель	Действие
1	ПК	ПК	LFS	Сборка кросс-компилятора cc1 с использованием cc-pc на ПК
2	ПК	LFS	LFS	Сборка компилятора cc-lfs с использованием cc1 на ПК
3	LFS	LFS	LFS	Пересборка и тестирование cc-lfs, используя cc-lfs в lfs

В приведенной выше таблице «ПК» означает, что команды выполняются на компьютере с использованием уже установленного дистрибутива. «В lfs» означает, что команды выполняются в chroot-окружении.

Это еще не конец истории. Язык C - это не просто компилятор; также он определяет стандартную библиотеку. В этой книге используется библиотека GNU C под названием glibc (есть альтернативный вариант - "musl"). Эта библиотека должна быть скомпилирована для машины lfs, то есть с использованием кросс-компилятора cc1. Но сам компилятор использует внутреннюю библиотеку, реализующую сложные инструкции, недоступные в наборе инструкций ассемблера. Эта внутренняя библиотека называется libgcc, и для полноценной работы ее необходимо связать с библиотекой glibc! Кроме того, стандартная библиотека для C++ (libstdc++) также должна быть связана с glibc. Решение этой проблемы курицы и яйца состоит в том, чтобы сначала собрать деградированную libgcc на основе cc1, в которой отсутствуют некоторые функциональные возможности, такие как потоки и обработка исключений, затем собрать glibc с использованием этого деградированного компилятора (сама glibc не деградирована), а затем собрать libstdc ++. В этой последней библиотеке будет не хватать некоторых функциональных возможностей libgcc.

Выводом из предыдущего абзаца является то, что cc1 не может собрать полнофункциональную libstdc++ с деградированной libgcc, но это единственный компилятор, доступный для сборки библиотек C/C++ на этапе 2. Есть две причины, по которым мы не используем сразу компилятор cc-lfs, собранный на этапе 2, для сборки этих библиотек.

- Вообще говоря, cc-lfs не может работать на ПК (хост-системе). Хотя триплеты для ПК и LFS совместимы друг с другом, исполняемый файл для lfs должен зависеть от glibc-2.40; хост-дистрибутив может использовать либо другую реализацию libc (например, musl), либо предыдущий выпуск glibc (например, glibc-2.13).
- Даже если cc-lfs может работать на ПК, его использование на ПК сопряжено с риском привязки к библиотекам ПК, так как cc-lfs является родным компилятором.

Поэтому, когда мы собираем gcc этап 2, мы даем указание системе сборки пересобрать libgcc и libstdc++ с помощью cc1, но мы связываем libstdc++ с новой пересобранной libgcc вместо старой, деградированной. Это делает пересобранную библиотеку libstdc++ полностью функциональной.

В Глава 8 (или «этап 3») собраны все пакеты, необходимые для системы LFS. Даже если пакет уже был установлен в системе LFS в предыдущей главе, мы все равно пересобираем пакет. Основная причина пересборки этих пакетов состоит в том, чтобы сделать их стабильными: если мы переустанавливаем пакет LFS в готовой системе LFS, содержимое пакета должно совпадать с содержимым того же пакета при первой установке в Глава 8. Временные пакеты, установленные в Глава 6 или Глава 7 не могут удовлетворять этому требованию, потому что некоторые из них собраны без необязательных зависимостей и autoconf не может выполнить некоторые проверки функций в Глава 6 из-за кросс-компиляции, в результате чего во временных пакетах отсутствуют дополнительные функции или используются не оптимальные процедуры кода. Кроме того, второстепенной причиной для пересборки пакетов является выполнение тестов.

Другие детали процесса

Кросс-компилятор будет установлен в отдельный каталог `$LFS/tools` , так как он не будет частью конечной системы.

Сначала устанавливается Binutils, потому что во время выполнения команды **configure** gcc и glibc выполняются различные тесты функций на ассемблере и компоновщике, чтобы определить, какие программные функции следует включить или отключить. Это важнее, чем может показаться на первый взгляд. Неправильно настроенный gcc или glibc может привести к незначительной поломке сборочных инструментов, где последствия такой поломки могут проявиться ближе к концу сборки всего дистрибутива. Сбой тестов обычно выявляет эту ошибку до того, как будет выполнено много дополнительной работы.

Binutils устанавливает свой ассемблер и компоновщик в двух местах: `$LFS/tools/bin` и `$LFS/tools/$LFS_TGT/bin` . Инструменты в одном месте жестко связаны с другими. Важным аспектом компоновщика является порядок поиска в библиотеке. Подробную информацию можно получить от **ld**, передав ей флаг -

-verbose . Например, **$LFS_TGT-ld --verbose | grep SEARCH** покажет текущие пути поиска и их порядок. Он показывает, какие файлы связаны с помощью **ld**, путем компиляции фиктивной программы и передачи параметра *--verbose* компоновщику. Например, **$LFS_TGT-gcc dummy.c -Wl,--verbose 2>&1 | grep succeeded** покажет все файлы, успешно открытые во время компоновки.

Следующий устанавливаемый пакет — gcc. Пример того, что можно увидеть во время запуска **configure**:

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

Это важно по причинам, упомянутым выше. Также здесь демонстрируется, что сценарий настройки gcc не просматривает значения переменной PATH, чтобы найти, какие инструменты использовать. Однако во время фактической работы самого **gcc** не обязательно используются одни и те же пути поиска. Чтобы узнать, какой стандартный компоновщик будет использовать **gcc**, запустите: **$LFS_TGT-gcc -print-prog-name=ld**.

Подробную информацию можно получить из **gcc**, передав ему параметр *-v* при компиляции фиктивной программы. Например, **gcc -v dummy.c** покажет подробную информацию об этапах препроцессора, компиляции и сборки, включая указанные в **gcc** пути поиска и их порядок.

Далее устанавливаются очищенные заголовочные файлы Linux API. Они позволяют стандартной библиотеке C (Glibc) взаимодействовать с функциями, предоставляемыми ядром Linux.

Следующий устанавливаемый пакет — glibc. Наиболее важными при сборке glibc являются компилятор, бинарные инструменты и заголовочные файлы ядра. С компилятором и бинарными инструментами, как правило, не бывает проблем, поскольку glibc всегда использует параметры, передаваемые скрипту configure, которые указаны в *--host* , например, в нашем случае компилятором будет **$LFS_TGT-gcc**, а инструментом **readelf** будет **$LFS_TGT-readelf**. С заголовочными файлами ядра может быть немного сложнее. Поэтому мы не рискуем и используем доступный параметр configure, чтобы обеспечить правильный выбор. После выполнения команды **configure** проверьте содержимое файла `config.make` в каталоге `build` на наличие всех нужных параметров. Эти элементы подчеркивают важный аспект пакета glibc — он очень самодостаточен с точки зрения своего механизма сборки и, как правило, не полагается на значения по умолчанию.

Как было сказано выше, затем компилируется стандартная библиотека C++, а затем в Глава 6 все остальные программы, которым необходимо разрешить проблему циклических зависимостей во время сборки. На этапе установки всех этих пакетов используется переменная DESTDIR, для принудительной установки в файловую систему LFS.

В конце Глава 6 устанавливается собственный компилятор lfs. Сначала собирается binutils с той же переменной `DESTDIR`, что и другие программы, затем повторно собирается gcc, без сборки некоторых некритических библиотек. Из-за какой-то странной логики в сценарии настройки GCC `CC_FOR_TARGET` заканчивается как **cc**, когда хост совпадает с целью, но отличается от системы сборки. Поэтому значение *CC_FOR_TARGET=$LFS_TGT-gcc* явно указывается в параметрах конфигурации.

После входа в среду chroot в Глава 7 первой задачей является установка libstdc++. Затем выполняется установка временных программ, необходимых для правильной работы тулчейна. С этого момента основной набор инструментов является самодостаточным и автономным. В Глава 8 собираются, тестируются и устанавливаются окончательные версии всех пакетов, необходимых для полнофункциональной системы.

Общие инструкции по компиляции

Внимание

В процессе разработки LFS инструкции в книге часто изменяются, чтобы адаптироваться к обновлению пакета или использовать преимущества новых функций из обновленных пакетов. Смешение инструкций разных версий книги LFS может привести к незначительным поломкам. Такого рода проблемы обычно являются результатом повторного использования некоторых скриптов, созданных для предыдущей версии LFS. Такое повторное использование настоятельно не рекомендуется. Если вы по какой-либо причине повторно используете скрипты из предыдущей версии LFS, вам нужно быть очень осторожным при обновлении скриптов, чтобы они соответствовали текущей версии книги LFS.

При сборке пакетов в инструкциях делается несколько допущений:

- На некоторые пакеты необходимо наложить патчи перед компиляцией, метод используется тогда, когда исправление необходимо для решения проблем сборки. Патчи часто требуются как в этой, так и в следующих главах, но иногда, когда один и тот же пакет собирается более одного раза, патч требуется не сразу. Поэтому не беспокойтесь, если инструкции для скачанного патча отсутствуют. Предупреждающие сообщения о *смещении (offset)* или *размытии (fuzz)* также могут появляться при применении патча. Не обращайте внимания на эти предупреждения, патч все равно успешно применен.
- Во время компиляции большинства пакетов на экране будут отображаться предупреждения. Это нормально, и их можно смело игнорировать. Предупреждения появляются, например, когда используется устаревший, недопустимый синтаксис C или C++. Стандарты C меняются довольно часто, и некоторые пакеты все еще используют более старый стандарт. Это не является серьезной проблемой, но вызывает появление предупреждений.
- Проверьте в последний раз, что переменная среды `LFS` настроена правильно:

```
echo $LFS
```

  Убедитесь, что в выводе указан путь к точке монтирования раздела LFS, то есть `/mnt/lfs` , как в примере из этой книги.

- Наконец, необходимо подчеркнуть два важных момента:

Важно

Инструкции по сборке предполагают, что все Требования к хост-системе, включая символические ссылки, установлены правильно:

- **bash** это используемая оболочка.
- **sh** это символическая ссылка на **bash**.
- **/usr/bin/awk** это символическая ссылка на **gawk**.
- **/usr/bin/yacc** это символическая ссылка на **bison** или небольшой скрипт, который выполняет bison

Важно

Вот краткое описание процесса сборки:

1. Поместите все исходники и патчи в каталог, который будет доступен из среды chroot, например, `/mnt/lfs/sources/` .
2. Перейдите в каталог `/mnt/lfs/sources/` .
3. Для каждого пакета:
   a. С помощью программы **tar** извлеките пакет для сборки. В Глава 5 и Глава 6 убедитесь, что при извлечении пакета вы залогинены под пользователем lfs.

      Не используйте никаких методов, кроме команды **tar**, для извлечения исходного кода. Примечательно, что использование команды **cp -R** для копирования дерева исходного кода в другое место может привести к уничтожению меток времени в дереве исходного кода и привести к сбою сборки.
   b. Перейдите в каталог, созданный при извлечении пакета.
   c. Следуйте инструкциям по сборке пакета.
   d. Вернитесь в исходный каталог, когда сборка будет завершена.
   e. Удалите извлеченный каталог, если не указано иное.

Глава 5. Сборка кросс-тулчейна

5.1. Введение

В этой главе дано описание, как создать кросс-компилятор и связанные с ним инструменты. Несмотря на то, что на данном этапе кросс-компиляция имитируется, принципы его работы те же, что и для настоящего кросс-тулчейна.

Программы, скомпилированные в этой главе, будут установлены в каталог `$LFS/tools` , чтобы они были отделены от файлов, установленных в следующих главах. Библиотеки, же, устанавливаются на свое постоянное место, поскольку они относятся к системе, которую мы хотим создать.

5.2. Binutils-2.43.1 - Проход 1

Пакет Binutils содержит компоновщик, ассемблер и другие инструменты для работы с объектными файлами.

Приблизительное время сборки:	1 SBU
Требуемое дисковое пространство:	677 MB

5.2.1. Установка кросс-пакета Binutils

Примечание

Вернитесь назад и перечитайте примечания в разделе Общие инструкции по компиляции. Понимание информации, помеченной как важная, может впоследствии избавить вас от многих проблем.

Очень важно, чтобы Binutils был скомпилированным первым, потому что и Glibc, и GCC выполняют различные тесты на доступных компоновщике и ассемблере, чтобы определить, какие из их функций следует включить.

В документации пакета Binutils рекомендуется выполнять сборку в отдельном каталоге, создадим его:

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

Примечание

Для того, чтобы значения SBU, перечисленные в остальной части книги, были вам полезны, измерьте время, необходимое для сборки этого пакета, начиная с настройки и заканчивая установкой. Чтобы добиться этого, оберните команды сборки командой **time**: `time { ../configure ... && make && make install; }`.

Теперь подготовьте Binutils к компиляции:

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

Значение параметров настройки:

`--prefix=$LFS/tools`
Указывает сценарию configure подготовить к установке пакет Binutils в каталог `$LFS/tools` .

`--with-sysroot=$LFS`
Для кросс-компляции указывает системе сборки искать в $LFS библиотеки целевой системы, если необходимо.

`--target=$LFS_TGT`
Поскольку название машины в значении переменной `LFS_TGT` может отличаться от значения, которое возвращает сценарий **config.guess**, этот аргумент укажет сценарию **configure** как настроить систему сборки пакета Binutils для создания кросс-компоновщика.

--disable-nls

Этот параметр отключает интернационализацию, так как i18n не требуется для временных инструментов.

--enable-gprofng=no

Этот параметр отключает сборку gprofng, который не нужен для временного инструментария.

--disable-werror

Этот параметр предотвращает остановку сборки в случае появления предупреждений от компилятора хоста.

--enable-new-dtags

Этот параметр заставляет компоновщик использовать тег «runpath» для встраивания путей поиска библиотек в исполняемые файлы и общие библиотеки вместо традиционного тега «rpath». Это упрощает отладку динамически подключаемых исполняемых файлов и устраняет потенциальные проблемы в наборах тестов некоторых пакетов.

--enable-default-hash-style=gnu

По умолчанию компоновщик генерирует как хеш-таблицу в стиле GNU, так и классическую хеш-таблицу ELF для общих библиотек и динамически связанных исполняемых файлов. Хеш-таблицы необходимы только для динамического компоновщика, выполняющего поиск символов. В LFS динамический компоновщик (предоставляемый пакетом Glibc) всегда будет использовать хеш-таблицу в стиле GNU, к которой запросы выполняются быстрее. Так что классическая хеш-таблица ELF совершенно бесполезна. Этот параметр указывает компоновщику по умолчанию генерировать только хеш-таблицу в стиле GNU, поэтому мы можем избежать траты времени на создание классической хеш-таблицы ELF при сборке пакетов или не тратить дисковое пространство для ее хранения.

--enable-multilib

Включает поддержку multilib.

Скомпилируйте пакет:

```
make
```

Установите пакет:

```
make install
```

Подробная информация об этом пакете находится в Раздел 8.20.2, «Содержимое пакета Binutils.»

5.3. GCC-14.2.0 - Проход 1

Пакет GCC содержит коллекцию компиляторов GNU, которая включает компиляторы C и C++.

Приблизительное время сборки:	3.2 SBU
Требуемое дисковое пространство:	4.9 GB

5.3.1. Установка кросс-пакета GCC

Для GCC требуются пакеты GMP, MPFR и MPC. Поскольку эти пакеты могут отсутствовать в дистрибутиве вашего хоста, они будут собраны с помощью GCC. Распакуйте каждый пакет в исходный каталог GCC и переименуйте получившиеся каталоги, чтобы процедуры сборки GCC использовали их автоматически:

Примечание

В этой главе часто возникают недоразумения, хотя применяются те же процедуры, что и в любой другой главе, следуйте инструкции которую получили ранее (Инструкции по сборке пакетов). Сначала распакуйте пакет gcc-14.2.0 из архива, а затем перейдите в созданный каталог. Только после этого следует приступить к приведенным ниже инструкциям.

```
tar -xf ../mpfr-4.2.1.tar.xz
mv -v mpfr-4.2.1 mpfr
tar -xf ../gmp-6.3.0.tar.xz
mv -v gmp-6.3.0 gmp
tar -xf ../mpc-1.3.1.tar.gz
mv -v mpc-1.3.1 mpc
```

Измените имя каталога по умолчанию для библиотек:

```
sed -e '/m64=/s/lib64/lib/' \
    -e '/m32=/s/m32=.*/m32=..\/lib32$(call if_multiarch,:i386-linux-gnu)/' \
    -i.orig gcc/config/i386/t-linux64
```

В документации к GCC рекомендуется собирать GCC в отдельном каталоге:

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

Подготовьте GCC к компиляции:

```
mlist=m64,m32
../configure                  \
    --target=$LFS_TGT                           \
    --prefix=$LFS/tools                         \
    --with-glibc-version=2.40                   \
    --with-sysroot=$LFS                         \
    --with-newlib                               \
    --without-headers                           \
    --enable-default-pie                        \
    --enable-default-ssp                        \
    --enable-initfini-array                     \
    --disable-nls                               \
    --disable-shared                            \
    --enable-multilib --with-multilib-list=$mlist  \
    --disable-decimal-float                     \
    --disable-threads                           \
    --disable-libatomic                         \
    --disable-libgomp                           \
    --disable-libquadmath                       \
    --disable-libssp                            \
    --disable-libvtv                            \
    --disable-libstdcxx                         \
    --enable-languages=c,c++
```

Значение параметров настройки:

--with-glibc-version=2.40

Этот параметр указывает версию Glibc, которая будет использоваться на целевой системе. Он не имеет отношения к libc хост-дистрибутива, потому что все, скомпилированное в этом разделе, будет выполняться в среде chroot, которая изолирована от libc хост-дистрибутива.

--with-newlib

Поскольку работающая библиотека C еще недоступна, это гарантирует, что константа inhibit_libc будет определена при сборке libgcc. Это предотвращает компиляцию любого кода, требующего поддержки libc.

--without-headers

При создании полного кросс-компилятора GCC требует наличия стандартных заголовков, совместимых с целевой системой. Для наших целей эти заголовки не понадобятся. Этот параметр предотвращает их поиск GCC.

--enable-default-pie и --enable-default-ssp

Эти параметры позволяют GCC по умолчанию компилировать программы с некоторыми функциями усиливающими безопасность (более подробная информация о них приведена в примечание о PIE и SSP в Главе 8). На данном этапе они не является строго обязательными, поскольку компилятор будет создавать только временные исполняемые файлы. Но лучше, чтобы временные пакеты были максимально приближены к тем, что будут в готовой системе LFS.

--disable-shared

Этот параметр заставляет GCC статически связывать свои внутренние библиотеки. Он необходим потому что общие библиотеки требуют Glibc, который еще не установлен в целевой системе.

--enable-multilib --with-multilib-list=...

LFS можно использовать с поддержкой multilib для архитектур которые указаны в списке with-multilib-list.

--disable-threads, --disable-libatomic, --disable-libgomp, --disable-libquadmath, --disable-libssp, --disable-libvtv, --disable-libstdcxx

Эти аргументы отключают поддержку расширений для работы с многопоточностью, libatomic, libgomp, libquadmath, libssp, libvtv и стандартной библиотеки C++ соответственно. Эти функции могут не

скомпилироваться при сборке кросс-компилятора и не нужны для задач кросс-компиляции временной libc

--enable-languages=c,c++

Этот параметр обеспечивает сборку только компиляторов C и C++. Это единственные языки, которые нужны сейчас.

Скомпилируйте GCC, выполнив:

```
make
```

Установите пакет:

```
make install
```

Во время сборки GCC установил пару внутренних системных заголовочных файлов. Обычно один из файлов `limits.h` , включает соответствующие системные ограничения `limits.h` , в данном случае `$LFS/usr/include/limits.h` . Однако во время сборки GCC `$LFS/usr/include/limits.h` не существует, поэтому только что установленный внутренний заголовочный файл является частичным, автономным файлом и не включает расширенные функции системного файла. Этого достаточно для сборки Glibc, но полный внутренний заголовочный файл понадобится позже. Создайте полную версию внутреннего заголовочного файла с помощью команды, идентичной той, что система сборки GCC использует обычно:

Примечание

В приведенной ниже команде показан пример подстановки вложенных команд, используя два метода: обратные кавычки и конструкцию `$()`. Его можно было бы переписать, используя один и тот же метод для обеих замен, но сделано так, чтобы продемонстрировать, как их можно использовать одновременно. В целом метод `$()` предпочтительнее.

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

Подробная информация об этом пакете находится в Раздел 8.30.2, «Содержимое пакета GCC.»

5.4. Заголовочные файлы Linux-6.10.5 API

Заголовочные файлы Linux API (в linux-6.10.5.tar.xz) предоставляют API ядра для использования Glibc.

Приблизительное время сборки: менее 0.1 SBU
Требуемое дисковое пространство: 1.6 GB

5.4.1. Установка заголовочных файлов

Ядро Linux должно предоставлять интерфейс прикладного программирования (API) для использования системной библиотекой C (Glibc в LFS). Это делается путем установки заголовочных файлов C, которые поставляются в архиве с исходным кодом ядра Linux.

Убедитесь, что в пакете нет устаревших файлов:

```
make mrproper
```

Теперь извлеките видимые пользователю заголовочные файлы ядра из исходного кода. Рекомендуемый способ make «headers_install» использовать нельзя, так как для этого требуется rsync, который может быть недоступен. Заголовочные файлы сначала помещаются в `/usr` , а затем копируются в нужное место.

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

5.4.2. Содержимое заголовочных файлов Linux API

Установленные заголовочные файлы:: /usr/include/asm/*.h, /usr/include/asm-generic/*.h, /usr/include/drm/*.h, /usr/include/linux/*.h, /usr/include/misc/*.h, /usr/include/mtd/*.h, /usr/include/rdma/*.h, /usr/include/scsi/*.h, /usr/include/sound/*.h, /usr/include/video/*.h, and /usr/include/xen/*.h
Созданные каталоги:: /usr/include/asm, /usr/include/asm-generic, /usr/include/drm, /usr/include/linux, /usr/include/misc, /usr/include/mtd, /usr/include/rdma, /usr/include/scsi, /usr/include/sound, /usr/include/video, and /usr/include/xen

Краткое описание

`/usr/include/asm/*.h`	Заголовочные файлы Linux API ASM
`/usr/include/asm-generic/*.h`	Заголовочные файлы Linux API ASM Generic
`/usr/include/drm/*.h`	Заголовочные файлы Linux API DRM
`/usr/include/linux/*.h`	Заголовочные файлы Linux API Linux
`/usr/include/misc/*.h`	Заголовочные файлы Linux API Miscellaneous
`/usr/include/mtd/*.h`	Заголовочные файлы API MTD
`/usr/include/rdma/*.h`	Заголовочные файлы Linux API RDMA
`/usr/include/scsi/*.h`	Заголовочные файлы Linux API SCSI
`/usr/include/sound/*.h`	Заголовочные файлы Linux API Sound
`/usr/include/video/*.h`	Заголовочные файлы Linux API Video
`/usr/include/xen/*.h`	Заголовочные файлы Linux API Xen

5.5. Glibc-2.40

Пакет Glibc содержит основную библиотеку C. Эта библиотека предоставляет основные процедуры для выделения памяти, поиска в каталогах, открытия и закрытия файлов, чтения и записи файлов, обработки строк, сопоставления с образцом, арифметики и так далее

Приблизительное время сборки: 1.3 SBU
Требуемое дисковое пространство: 828 MB

5.5.1. Установка пакета Glibc

Во-первых, создайте символическую ссылку для соответствия требованиям LSB. Кроме того, для совместимости с x86_64 создайте символическую ссылку, необходимую для правильной работы загрузчика динамической библиотеки:

```
ln -sfv ../lib/ld-linux-x86-64.so.2 $LFS/lib64
ln -sfv ../lib/ld-linux-x86-64.so.2 $LFS/lib64/ld-lsb-x86-64.so.3
```

Примечание

Приведенная выше команда верна. Команда **ln** имеет несколько вариантов синтаксиса, поэтому обязательно ознакомьтесь с **info coreutils ln** и *ln(1)*, прежде чем сообщать об ошибке.

Некоторые программы, использующие Glibc, применяют несовместимый с FHS каталог `/var/db` для хранения своих данных времени выполнения. Установите следующий патч, чтобы такие программы хранили свои данные в местах, совместимых с FHS:

```
patch -Np1 -i ../glibc-2.40-fhs-1.patch
```

В документации к Glibc рекомендуется собирать Glibc в отдельном каталоге:

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

Убедитесь, что утилиты **ldconfig** and **sln** установлены в `/usr/sbin` :

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

Затем подготовьте Glibc к компиляции:

```
../configure                             \
      --prefix=/usr                      \
      --host=$LFS_TGT                    \
      --build=$(../scripts/config.guess) \
      --enable-kernel=4.19               \
      --with-headers=$LFS/usr/include    \
      --disable-nscd                     \
      libc_cv_slibdir=/usr/lib
```

Значение параметров настройки:

`--host=$LFS_TGT, --build=$(../scripts/config.guess)`
Комбинация этих опций указывает на то, что система сборки Glibc настраивается на кросс-компиляцию с использованием кросс-компоновщика и кросс-компилятора в `$LFS/tools` .

`--enable-kernel=4.19`
Этот параметр позволяет Glibc выполнять компиляцию библиотеки с поддержкой ядра 4.19 и более поздних версий. Поддержка более старых ядер не включена.

`--with-headers=$LFS/usr/include`

Этот аргумент позволяет скомпилировать библиотеку с заголовочными файлами, недавно установленными в каталоге $LFS/usr/include, таким образом, пакету будет известно, какие функции есть у ядра, чтобы оптимизировать себя.

`libc_cv_slibdir=/usr/lib`

Этот аргумент гарантирует, что библиотека будет установлена в /usr/lib вместо стандартного /lib64 на 64-битных машинах.

`--disable-nscd`

Параметр отключает сборку демона кэша службы имен, который больше не используется.

На этом этапе может появиться следующее предупреждение:

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

Отсутствующая или несовместимая программа **msgfmt**, как правило, безвредна. **msgfmt** является частью пакета Gettext, который должен предоставлять хост-дистрибутив.

Примечание

Поступали сообщения о том, что этот пакет может не компилироваться при «параллельной сборке». Если это произойдет, повторно запустите команду make с параметром `-j1 .`

Скомпилируйте пакет:

```
make
```

Установите пакет:

Предупреждение

Если переменная `LFS` настроена неправильно, и, несмотря на рекомендации, вы выполняете сборку от имени пользователя `root`, следующая команда установит только что собранный Glibc в вашу хост-систему, что, скорее всего, сделает её непригодной для использования. Поэтому дважды проверьте, правильность настройки среды и что вы вошли в систему не под учетной записью `root`, прежде чем запускать следующую команду.

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

Значение опции make install:

`DESTDIR=$LFS`

Переменная make `DESTDIR` используется почти всеми пакетами для определения места установки пакета. Если она не задана, по умолчанию для установки используется корневой каталог (`/`). Здесь мы указываем, что пакет должен быть установлен в $LFS, который станет корневым каталогом в Раздел 7.4, «Вход в окружение Chroot».

Исправьте жестко запрограммированный путь к исполняемому загрузчику в **ldd**:

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

Внимание

На этом этапе необходимо остановиться и убедиться, что основные функции (компиляция и компоновка) нового кросс-тулчейна работают должным образом. Чтобы выполнить проверку работоспособности, выполните следующие команды:

```
echo 'int main(){}' | $LFS_TGT-gcc -xc -
readelf -l a.out | grep ld-linux
```

Если все работает правильно, ошибок быть не должно и вывод последней команды будет иметь вид:

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

Если выходные данные отображаются не так, как указано выше, или их вообще нет, значит, что-то сделано неправильно. Разберитесь с проблемой и повторите шаги выше, чтобы исправить ее. Эта проблема должна быть решена, прежде чем вы продолжите.

Как только все будет хорошо, удалите тестовый файл:

```
rm -v a.out
```

Примечание

Сборка пакетов в следующей главе послужит дополнительной проверкой правильности сборки временного кросс-тулчейна. Если какой-либо пакет, особенно Binutils или GCC, не удается собрать, это указывает на то, что что-то пошло не так с установленными ранее Binutils, GCC, или Glibc.

5.5.2. Сборка Glibc — 32-битная версия

Теперь перекомпилируйте Glibc для m32. Извлеченный исходный код можно использовать повторно, но его необходимо очистить перед установкой версии Glibc m32.

Очистите каталог сборки и удалите артефакты от предыдущей сборки:

```
make clean
find .. -name "*.a" -delete
```

Сконфигурируйте Glibc для m32 с помощью следующих команд:

```
CC="$LFS_TGT-gcc -m32" \
CXX="$LFS_TGT-g++ -m32" \
../configure                             \
      --prefix=/usr                      \
      --host=$LFS_TGT32                  \
      --build=$(../scripts/config.guess) \
      --enable-kernel=4.19                \
      --with-headers=$LFS/usr/include    \
      --disable-nscd                     \
      --libdir=/usr/lib32                \
      --libexecdir=/usr/lib32            \
      libc_cv_slibdir=/usr/lib32
```

Скомпилируйте пакет:

```
make
```

Установите пакет:

```
make DESTDIR=$PWD/DESTDIR install
cp -a DESTDIR/usr/lib32 $LFS/usr/
install -vm644 DESTDIR/usr/include/gnu/{lib-names,stubs}-32.h \
               $LFS/usr/include/gnu/
ln -svf ../lib32/ld-linux.so.2 $LFS/lib/ld-linux.so.2
```

Внимание

На этом этапе необходимо остановиться и убедиться, что основные функции (компиляция и компоновка) нового кросс-тулчейна работают должным образом. Чтобы выполнить проверку работоспособности, выполните следующие команды:

```
echo 'int main(){}' > dummy.c
$LFS_TGT-gcc -m32 dummy.c
readelf -l a.out | grep '/ld-linux'
```

Если все работает правильно, ошибок быть не должно и вывод последней команды будет иметь вид:

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

Если выходные данные отображаются не так, как указано выше, или их вообще нет, значит, что-то сделано неправильно. Разберитесь с проблемой и повторите шаги выше, чтобы исправить ее. Эта проблема должна быть решена, прежде чем вы продолжите.

Как только все будет хорошо, удалите тестовый файл:

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

Подробная информация об этом пакете находится в Раздел 8.5.4, «Содержимое пакета Glibc.»

5.6. Libstdc++ из GCC-14.2.0

Libstdc++ — это стандартная библиотека C++. Она нужна для компиляции кода C++ (часть GCC написана на C++), когда мы собирали GCC-Проход 1, нам пришлось отложить её установку, потому что она зависит от библиотеки Glibc, которой еще не было в целевом каталоге.

Приблизительное время сборки: 0.2 SBU
Требуемое дисковое пространство: 1.2 GB

5.6.1. Установка библиотеки Libstdc++

Примечание

Libstdc++ является частью исходников GCC. Сначала вы должны распаковать архив GCC и перейти в каталог `gcc-14.2.0` .

Создайте отдельный каталог сборки для libstdc++ и перейдите в него:

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

Подготовьте libstdc++ к компиляции:

```
../libstdc++-v3/configure           \
    --host=$LFS_TGT                 \
    --build=$(../config.guess)      \
    --prefix=/usr                   \
    --enable-multilib               \
    --disable-nls                   \
    --disable-libstdcxx-pch         \
    --with-gxx-include-dir=/tools/$LFS_TGT/include/c++/14.2.0
```

Значение параметров настройки:

`--host=...`
Указывает, что должен использоваться кросс-компилятор, который мы только что собрали, вместо того, который находится в `/usr/bin` .

`--enable-multilib`
Включает сборку объектов multilib.

`--disable-libstdcxx-pch`
Этот аргумент предотвращает установку предварительно скомпилированных include-файлов, которые на данном этапе не нужны.

`--with-gxx-include-dir=/tools/$LFS_TGT/include/c++/14.2.0`
Указывает каталог установки для include-файлов. Поскольку libstdc++ является стандартной библиотекой C++ для LFS, этот каталог должен соответствовать местоположению, в котором компилятор C++ (**$LFS_TGT-g++**) будет искать стандартные включаемые файлы C++. При обычной сборке эта информация автоматически передается в Libstdc++ при выполнении **configure** из каталога верхнего уровня. В нашем случае эта информация должна быть указана явно. Компилятор C++ добавит путь sysroot `$LFS` (указанный при сборке GCC Проход 1) к пути поиска include-файлов, поэтому фактически он будет искать в `$LFS/tools/$LFS_TGT/include/c++/14.2.0` . Комбинация переменной *`DESTDIR`* (в приведенной ниже команде **make install**) и этого аргумента обеспечивает установку заголовочных файлов туда.

Скомпилируйте Libstdc++, выполнив:

```
make
```

Установите библиотеку:

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

Удалите архивные файлы libtool, поскольку они потенциально опасны при кросс-компиляции:

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

Подробная информация об этом пакете приведена в Раздел 8.30.2, «Содержимое пакета GCC.»

Глава 6. Кросс-Компиляция временных инструментов

6.1. Введение

В этой главе рассказывается, как выполнить кросс-компиляцию базовых утилит с использованием только что собранного кросс-тулчейна. Эти утилиты установлены в свое конечное местоположение, но пока не могут быть использованы. Выполняемые инструкции по-прежнему зависят от инструментария хоста. Тем не менее, установленные библиотеки используются при компоновке.

Использование утилит станет возможным в следующей главе после входа в среду «chroot». Все пакеты из этой главы, должны быть собраны до того, как мы это сделаем. Поэтому пока наша система зависима от хост-системы.

Еще раз напомним, что неправильная настройка `LFS` вместе со сборкой от `root` может сделать ваш компьютер непригодным для использования. Всю эту главу нужно выполнить от имени пользователя `lfs`, в его рабочем окружении, как описано в Раздел 4.4, «Настройка окружения».

6.2. M4-1.4.19

Пакет M4 содержит макропроцессор.

Приблизительное время сборки:	0.1 SBU
Требуемое дисковое пространство:	31 MB

6.2.1. Установка пакета M4

Подготовьте пакет M4 к компиляции:

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

Скомпилируйте пакет:

```
make
```

Установите пакет:

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

Подробная информация об этом пакете находится в Раздел 8.13.2, «Содержимое пакета M4.»

6.3. Ncurses-6.5

Пакет Ncurses содержит библиотеки для независимой от терминала обработки ввода/вывода

Приблизительное время сборки: 0.4 SBU
Требуемое дисковое пространство: 53 MB

6.3.1. Установка пакета Ncurses

Во-первых, убедитесь, что **gawk** найден первым во время настройки:

```
sed -i s/mawk// configure
```

Затем выполните следующие команды, чтобы собрать программу «tic» на хосте сборки:

```
mkdir build
pushd build
  ../configure
  make -C include
  make -C progs tic
popd
```

Подготовьте Ncurses к компиляции:

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

Значение новых параметров настройки:

`--with-manpage-format=normal`
Этот аргумент предотвращает установку Ncurses сжатых страниц руководства, это может произойти, если сам дистрибутив хоста содержит сжатые страницы руководства.

`--with-shared`
Этот аргумент позволяет Ncurses собирать и устанавливать разделяемые библиотеки C.

`--without-normal`
Этот аргумент предотвращает сборку и установку статических библиотек C.

`--without-debug`
Этот аргумент предотвращает сборку и установку отладочных библиотек.

`--with-cxx-shared`
Это аргумент позволяет Ncurses собирать и устанавливать общие привязки C++. А также предотвращает сборку и установку статических привязок C++.

`--without-ada`
Этот аргумент гарантирует, что Ncurses будет собран без поддержки компилятора Ada, который может присутствовать на хосте, но будет недоступен, как только мы войдем в среду **chroot**.

`--disable-stripping`
Этот аргумент не позволяет системе сборки использовать программу **strip** с хоста. Использование инструментов хоста в кросс-компилируемой программе может привести к сбою.

Скомпилируйте пакет:

```
make
```

Установите пакет:

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

Значение параметров установки:

TIC_PATH=$(pwd)/build/progs/tic
Нам нужно передать путь до только что собранной программы **tic**, которая работает на сборочной машине, чтобы база данных терминала была создана без ошибок.

ln -sv libncursesw.so $LFS/usr/lib/libncurses.so
Библиотека `libncurses.so` необходима для нескольких пакетов, которые мы скоро соберем. Мы создаем эту символическую ссылку, чтобы использовать `libncursesw.so` в качестве замены.

sed -e 's/^#if.*XOPEN.*$/#if 1/' ...
Заголовочный файл `curses.h` содержит определения различных структур данных Ncurses. С разными определениями макросов препроцессора могут использоваться два разных набора определений структуры данных: 8-битное определение совместимо с `libncurses.so` , а определение расширенного набора символов совместимо с `libncursesw.so` . Поскольку мы используем `libncursesw.so` вместо `libncurses.so` , отредактируйте заголовочный файл, чтобы он всегда использовал определение структуры данных расширенного набора символов, совместимое с `. libncursesw.so` .

6.3.2. Сборка Ncurses — 32-битная версия

Выполните очистку после предыдущей сборки:

```
make distclean
```

Подготовьте Ncurses к компиляции:

```
CC="$LFS_TGT-gcc -m32"              \
CXX="$LFS_TGT-g++ -m32"             \
./configure --prefix=/usr           \
            --host=$LFS_TGT32       \
            --build=$(./config.guess)    \
            --libdir=/usr/lib32     \
            --mandir=/usr/share/man \
            --with-shared           \
            --without-normal        \
            --with-cxx-shared       \
            --without-debug         \
            --without-ada           \
            --disable-stripping
```

Скомпилируйте пакет:

```
make
```

Установите пакет:

```
make DESTDIR=$PWD/DESTDIR TIC_PATH=$(pwd)/build/progs/tic install
ln -sv libncursesw.so DESTDIR/usr/lib32/libncurses.so
cp -Rv DESTDIR/usr/lib32/* $LFS/usr/lib32
rm -rf DESTDIR
```

Подробная информация об этом пакете находится в Раздел 8.31.3, «Содержимое пакета Ncurses.»

6.4. Bash-5.2.32

Пакет Bash содержит Bourne-Again Shell.

Приблизительное время сборки:	0.2 SBU
Требуемое дисковое пространство:	67 MB

6.4.1. Установка пакета Bash

Подготовьте Bash к компиляции:

```
./configure --prefix=/usr                      \
            --build=$(sh support/config.guess) \
            --host=$LFS_TGT                    \
            --without-bash-malloc              \
            bash_cv_strtold_broken=no
```

Значение параметров настройки:

`--without-bash-malloc`
: Этот параметр отключает использование функции распределения памяти (`malloc`) Bash, которая, как известно, вызывает ошибки сегментации. Если опция отключена, Bash будет использовать функции `malloc` из Glibc, которые более стабильны.

Скомпилируйте пакет:

```
make
```

Установите пакет:

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

Создайте символическую ссылку для программ, которые используют **sh** как оболочку:

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

Подробная информация об этом пакете находится в Раздел 8.37.2, «Содержимое пакета Bash.»

6.5. Coreutils-9.5

Пакет Coreutils содержит основные утилиты, необходимые каждой операционной системе.

Приблизительное время сборки: 0.3 SBU
Требуемое дисковое пространство: 175 MB

6.5.1. Установка пакета Coreutils

Подготовьте Coreutils к компиляции:

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

Значение параметров настройки:

`--enable-install-program=hostname`
Этот параметр позволяет создать и установить двоичный файл **hostname** – по умолчанию он отключен, но требуется для набора тестов Perl.

Скомпилируйте пакет:

```
make
```

Установите пакет:

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

Переместите программы в их конечное местоположение. Хотя во временной среде в этом нет необходимости, мы должны это сделать, потому что некоторые программы жестко прописывают местоположение исполняемых файлов:

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

Подробная информация об этом пакете находится в Раздел 8.59.2, «Содержимое пакета Coreutils.»

6.6. Diffutils-3.10

Пакет Diffutils содержит программы, которые показывают различия между файлами или каталогами.

Приблизительное время сборки:	0.1 SBU
Требуемое дисковое пространство:	29 MB

6.6.1. Установка пакета Diffutils

Подготовьте Diffutils для компиляции:

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

Скомпилируйте пакет:

```
make
```

Установите пакет:

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

Подробная информация об этом пакете находится в Раздел 8.61.2, «Содержимое пакета Diffutils.»

6.7. File-5.45

Пакет File содержит утилиту для определения типа указанного файла или файлов

Приблизительное время сборки: 0.1 SBU
Требуемое дисковое пространство: 37 MB

6.7.1. Установка пакета File

Команда **file** на хосте сборки должна быть той же версии, что и собираемая, чтобы создать файл подписи. Выполните следующие команды, чтобы создать временную копию команды **file**.

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

Значение новой опции настройки:

`--disable-*`
Сценарий конфигурации пытается использовать некоторые пакеты из основного дистрибутива, если существуют соответствующие файлы библиотек. Это может привести к сбою компиляции, если файлы библиотек существует, но отсутствуют соответствующие заголовочные файлы. Эти параметры предотвращают использование ненужных возможностей хоста.

Подготовьте файл для компиляции:

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

Скомпилируйте пакет:

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

Установите пакет:

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

Удалите архивный файл libtool, поскольку он потенциально опасен при кросс-компиляции:

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

Подробная информация об этом пакете находится в Раздел 8.11.3, «Содержимое пакета File.»

6.8. Findutils-4.10.0

Пакет Findutils содержит программы для поиска файлов. Эти программы предназначены для поиска по всем файлам в дереве каталогов, а также для создания, обслуживания и поиска в базе данных (часто быстрее, чем рекурсивный поиск, но ненадежно, если база данных давно не обновлялась).Findutils также предоставляет программу **xargs**, которую можно использовать для запуска указанной команды для каждого файла, выбранного при поиске.

Приблизительное время сборки: 0.2 SBU
Требуемое дисковое пространство: 48 MB

6.8.1. Установка пакета Findutils

Подготовьте Findutils к компиляции:

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

Скомпилируйте пакет:

```
make
```

Установите пакет:

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

Подробная информация об этом пакете находится в Раздел 8.63.2, «Содержимое пакета Findutils.»

6.9. Gawk-5.3.0

Пакет Gawk содержит программы для работы с текстовыми файлами.

Приблизительное время сборки:	0.1 SBU
Требуемое дисковое пространство:	47 MB

6.9.1. Установка пакета Gawk

Во-первых, убедитесь, что некоторые ненужные файлы не будут установлены:

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

Подготовьте Gawk к компиляции:

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

Скомпилируйте пакет:

```
make
```

Установите пакет:

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

Подробная информация об этом пакете находится в Раздел 8.62.2, «Содержимое пакета Gawk.»

6.10. Grep-3.11

Пакет Grep содержит программы для поиска по содержимому файлов.

Приблизительное время сборки: 0.1 SBU
Требуемое дисковое пространство: 27 MB

6.10.1. Установка пакета Grep

Подготовьте Grep к компиляции:

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

Скомпилируйте пакет:

```
make
```

Установите пакет:

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

Подробная информация об этом пакете находится в Раздел 8.36.2, «Содержимое пакета Grep.»

6.11. Gzip-1.13

Пакет Gzip содержит программы для сжатия и распаковки файлов.

Приблизительное время сборки:	0.1 SBU
Требуемое дисковое пространство:	11 MB

6.11.1. Установка пакета Gzip

Подготовьте Gzip к компиляции:

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

Скомпилируйте пакет:

```
make
```

Установите пакет:

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

Подробная информация об этом пакете находится в Раздел 8.66.2, «Содержимое пакета Gzip.»

6.12. Make-4.4.1

Пакет Make содержит программу, управляющую генерацией исполняемых и других файлов, из исходного кода.

Приблизительное время сборки: менее 0.1 SBU
Требуемое дисковое пространство: 15 MB

6.12.1. Установка пакета Make

Подготовьте Make к компиляции:

```
./configure --prefix=/usr   \
            --without-guile \
            --host=$LFS_TGT \
            --build=$(build-aux/config.guess)
```

Значение новой опции настройки:

`--without-guile`
Несмотря на то, что мы выполняем кросс-компиляцию, configure пытается использовать guile с узла сборки, если он его находит. Это приводит к сбою компиляции, этот аргумент предотвращает его использование.

Скомпилируйте пакет:

```
make
```

Установите пакет:

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

Подробная информация об этом пакете находится в Раздел 8.70.2, «Содержимое пакета Make.»

6.13. Patch-2.7.6

Пакет Patch содержит программу для изменения или создания файлов путём наложение «патча», обычно, создаваемого программой **diff**.

Приблизительное время сборки:	0.1 SBU
Требуемое дисковое пространство:	12 MB

6.13.1. Установка пакета Patch

Подготовьте Patch к компиляции:

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

Скомпилируйте пакет:

```
make
```

Установите пакет:

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

Подробная информация об этом пакете находится в Раздел 8.71.2, «Содержимое пакета Patch.»

6.14. Sed-4.9

Пакет Sed содержит потоковый редактор текста

Приблизительное время сборки:	0.1 SBU
Требуемое дисковое пространство:	21 MB

6.14.1. Установка пакета Sed

Подготовьте Sed к компиляции:

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

Скомпилируйте пакет:

```
make
```

Установите пакет:

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

Подробная информация об этом пакете находится в Раздел 8.32.2, «Содержимое пакета Sed.»

6.15. Tar-1.35

Пакет Tar предоставляет возможность создавать tar архивы, а также производить с ними различные манипуляции. Tar может распаковать предварительно созданный архив, добавить или обновить файлы в нём, вернуть список файлов в архиве.

Приблизительное время сборки: 0.1 SBU
Требуемое дисковое пространство: 42 MB

6.15.1. Установка пакета Tar

Подготовьте Tar к компиляции:

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

Скомпилируйте пакет:

```
make
```

Установите пакет:

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

Подробная информация об этом пакете находится в Раздел 8.72.2, «Содержимое пакета Tar.»

6.16. Xz-5.6.2

Пакет Xz содержит программы для сжатия и распаковки файлов. Он предоставляет возможности для lzma и более новых форматов сжатия xz. Сжатие текстовых файлов с помощью **xz** дает лучший процент сжатия, чем с традиционные **gzip** или **bzip2**.

Приблизительное время сборки: 0.1 SBU
Требуемое дисковое пространство: 20 MB

6.16.1. Установка пакета Xz

Подготовьте Xz к компиляции:

```
./configure --prefix=/usr                     \
            --host=$LFS_TGT                   \
            --build=$(build-aux/config.guess) \
            --disable-static                  \
            --docdir=/usr/share/doc/xz-5.6.2
```

Скомпилируйте пакет:

```
make
```

Установите пакет:

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

Удалите архивный файл libtool, поскольку он потенциально опасен при кросс-компиляции:

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

Подробная информация об этом пакете находится в Раздел 8.8.3, «Содержимое пакета Xz.»

6.17. Binutils-2.43.1 - Проход 2

Пакет Binutils содержит компоновщик, ассемблер и другие инструменты для работы с объектными файлами.

Приблизительное время сборки: 0.4 SBU
Требуемое дисковое пространство: 549 MB

6.17.1. Установка пакета Binutils

Система сборки Binutils использует содержащуюся в пакете копию libtool для линковки с внутренними статическими библиотеками, но копии libiberty и zlib, поставляемые с пакетом, не используют libtool. Это несоответствие может привести к тому, что созданные двоичные файлы будут ошибочно связаны с библиотеками из основного дистрибутива. Решение этой проблемы:

```
sed '6009s/$add_dir//' -i ltmain.sh
```

Создайте отдельный каталог для сборки:

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

Подготовьте Binutils к компиляции:

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

Значение новых параметров настройки:

`--enable-shared`
Собирает `libbfd` как разделяемую библиотеку

`--enable-64-bit-bfd`
Включает 64-разрядную поддержку (на хостах с меньшим размером слова). В 64-разрядных системах это может и не понадобиться, но вреда от этого не будет

`--enable-multilib`
Включает поддержку multilib в bintutils.

Скомпилируйте пакет:

```
make
```

Установите пакет:

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

Удалите архивные файлы libtool, поскольку они потенциально опасны при кросс-компиляции, также удалите ненужные статические библиотеки

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

Подробная информация об этом пакете находится в Раздел 8.20.2, «Содержимое пакета Binutils.»

6.18. GCC-14.2.0 - Проход 2

Пакет GCC содержит коллекцию компиляторов GNU, которая включает компиляторы C и C++.

Приблизительное время сборки:	4.2 SBU
Требуемое дисковое пространство:	5.5 GB

6.18.1. Установка пакета GCC

Как и при первой сборке GCC, требуются пакеты GMP, MPFR и MPC. Распакуйте архивы и переименуйте каталоги:

```
tar -xf ../mpfr-4.2.1.tar.xz
mv -v mpfr-4.2.1 mpfr
tar -xf ../gmp-6.3.0.tar.xz
mv -v gmp-6.3.0 gmp
tar -xf ../mpc-1.3.1.tar.gz
mv -v mpc-1.3.1 mpc
```

Измените имя каталога по умолчанию для библиотек:

```
sed -e '/m64=/s/lib64/lib/' \
    -e '/m32=/s/m32=.*/m32=..\/lib32$(call if_multiarch,:i386-linux-gnu)/' \
    -i.orig gcc/config/i386/t-linux64
```

Переопределите правила сборки заголовочных файлов libgcc и libstdc++, чтобы разрешить создание этих библиотек с поддержкой потоков POSIX:

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

Снова создайте отдельный каталог сборки:

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

Перед началом сборки GCC не забудьте отключить все переменные среды, которые переопределяют флаги оптимизации по умолчанию.

Теперь подготовьте GCC к компиляции:

```
mlist=m64,m32
../configure                                       \
    --build=$(../config.guess)                     \
    --host=$LFS_TGT                                \
    --target=$LFS_TGT                              \
    LDFLAGS_FOR_TARGET=-L$PWD/$LFS_TGT/libgcc      \
    --prefix=/usr                                  \
    --with-build-sysroot=$LFS                      \
    --enable-default-pie                           \
    --enable-default-ssp                           \
    --disable-nls                                  \
    --enable-multilib --with-multilib-list=$mlist  \
    --disable-libatomic                            \
    --disable-libgomp                              \
    --disable-libquadmath                          \
    --disable-libsanitizer                         \
    --disable-libssp                               \
    --disable-libvtv                               \
    --enable-languages=c,c++
```

Значение новых параметров настройки:

`--with-build-sysroot=$LFS`

Обычно, использование *`--host`* гарантирует, что для сборки GCC используется кросс-компилятор, и этот компилятор знает, что он должен искать заголовочные файлы и библиотеки в `$LFS`. Но сборочная система GCC использует другие инструменты, которые не знают об этом местоположении. Этот параметр необходим для того, чтобы они могли найти нужные файлы в `$LFS`, а не на хосте.

`--target=$LFS_TGT`

Поскольку мы выполняем кросс-компиляцию GCC, невозможно собрать целевые библиотеки (`libgcc` и `libstdc++`) с помощью двоичных файлов GCC, скомпилированных на этом шаге, эти двоичные файлы не будут работать на хост-дистрибутиве. Система сборки GCC по умолчанию попытается использовать компиляторы C и C++ хоста в качестве обходного пути. Сейчас не поддерживается создание целевых библиотек GCC с помощью другой версии GCC, поэтому использование компиляторов хоста может привести к сбою сборки. Этот параметр гарантирует сборку библиотек с помощью GCC собранного в предыдущей главе.

`LDFLAGS_FOR_TARGET=...`

Разрешить `libstdc++` использовать `libgcc`, собранную на этом этапе, вместо предыдущей версии, собранной в GCC-Проход 1. Предыдущая версия не поддерживает должным образом обработку исключений на C++, поскольку она была собрана без поддержки libc

`--disable-libsanitizer`

Отключает библиотеки среды выполнения GCC sanitizer. Они не нужны для временного набора инструментов. В GCC-Проход 1 это решалось с помощью параметра *`--disable-libstdcxx`* , но теперь мы должны передать его явно.

Скомпилируйте пакет:

```
make
```

Установите пакет:

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

В качестве завершающего штриха создайте символическую ссылку на утилиту. Многие программы и скрипты используют **cc** вместо **gcc**, чтобы сделать программы более универсальными и, следовательно, для совместимости со всеми типами UNIX-систем, где компилятор GNU C не всегда установлен. Наличие **cc** оставляет системному администратору право самостоятельно решать, какой компилятор C устанавливать:

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

Подробная информация об этом пакете находится в Раздел 8.30.2, «Содержимое пакета GCC.»

Глава 7. Вход в окружение Chroot и создание дополнительных временных инструментов

7.1. Введение

В этой главе рассказывается, как собрать последние недостающие части временной системы: инструменты, необходимые для сборки различных пакетов. Теперь, когда все циклические зависимости устранены, для сборки можно использовать среду «chroot», полностью изолированную от операционной системы хоста (за исключением работающего ядра).

Для правильной работы изолированной среды необходимо установить связь с работающим ядром. Это делается с помощью так называемых *виртуальных файловых систем ядра*, которые будут смонтированы перед входом в среду chroot. Вы можете проверить, смонтированы ли они, выполнив команду **findmnt**.

До Раздел 7.4, «Вход в окружение Chroot» команды должны выполняться от имени `root` с установленной переменной `LFS`. После входа в chroot все команды выполняются от имени `root`, к счастью, без доступа к операционной системе компьютера, на котором вы собираете LFS. В любом случае будьте осторожны, так как неверными командами легко разрушить всю систему LFS.

7.2. Смена владельца

Примечание

Команды, приведенные в оставшейся части книги, должны выполняться от имени пользователя `root`, а не `lfs`. Дважды проверьте, что переменная `$LFS` установлена в переменных окружения пользователя `root`.

В настоящее время вся иерархия каталогов в `$LFS` принадлежит пользователю `lfs`, существующему только на хост-системе. Если права на файлы и каталоги внутри `$LFS` оставить как есть, то они будут принадлежать ID пользователя без существующей учетной записи. Это опасно, так как созданная позже учетная запись, может получить такой же ID пользователя и стать владельцем всех файлов в `$LFS`, тем самым делая эти файлы уязвимыми для возможных злонамеренных манипуляций.

Для решения проблемы измените владельца каталогов `$LFS/*` на пользователя `root`, выполнив следующую команду:

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

Существует еще несколько каталогов, которые присутствуют в системах с поддержкой multilib. Измените их владельца тоже:

```
chown -R root:root $LFS/lib32
```

7.3. Подготовка виртуальных файловых систем ядра

Приложения, работающие в пользовательском пространстве, используют различные файловые системы, созданные ядром, для взаимодействия с самим ядром. Эти файловые системы являются виртуальными: для них не используется дисковое пространство. Содержимое файловых систем находится в памяти. Эти файловые системы должны быть смонтированы в дереве каталогов $LFS, чтобы приложения могли найти их в среде **chroot**.

Начните с создания каталогов, в которые будут смонтированы эти виртуальные файловые системы:

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

7.3.1. Монтирование и заполнение /dev

Во время обычной загрузки ядро автоматически монтирует файловую систему devtmpfs в каталог /dev ; ядро создает узлы устройств в этой виртуальной файловой системе в процессе загрузки или при первом обнаружении устройства, или доступе к нему. Демон udev может изменять владельца или разрешения узлов устройств, созданных ядром, или создавать новые узлы устройств или символические ссылки, чтобы облегчить работу разработчиков дистрибутива или системных администраторов. (Подробности смотрите в Раздел 9.3.2.2, «Создание узла устройства».) Если ядро хоста поддерживает devtmpfs, мы можем просто смонтировать devtmpfs в $LFS/dev и положиться на ядро для его заполнения.

Но в некоторых ядрах хоста отсутствует поддержка devtmpfs, эти хост-дистрибутивы используют разные методы для создания содержимого /dev . Таким образом, единственный независимый от хоста способ заполнить каталог $LFS/dev - это привязка к каталогу /dev хост-системы. Связное монтирование - это особый тип монтирования, который делает дерево каталога или файл видимым в каком-либо другом месте. Для этого используйте следующую команду:

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

7.3.2. Монтирование виртуальных файловых систем ядра

Теперь смонтируйте оставшиеся виртуальные файловые системы:

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

Значение параметров монтирования для devpts:

gid=5

Этот параметр гарантирует, что все узлы устройств, созданные devpts, принадлежат группе с идентификатором 5. Это идентификатор, который мы будем использовать позже для группы tty. Мы используем идентификатор группы вместо имени, поскольку хост-система может использовать другой идентификатор для своей группы tty.

mode=0620

Этот параметр гарантирует, что все узлы устройств, созданные devpts, будут иметь права 0620 (доступен для чтения и записи пользователем, доступен для записи группе). Вместе с вышеуказанной опцией это гарантирует, что devpts создаст узлы устройств, соответствующие требованиям grantpt(), а это означает, что вспомогательный файл Glibc **pt_chown** (который не установлен по умолчанию) не требуется.

В некоторых хост-системах /dev/shm является символической ссылкой на каталог /run/shm . /run tmpfs был смонтирован выше, поэтому сейчас необходимо только создать каталог с правильными разрешениями.

В других хост-системах /dev/shm является точкой монтирования для tmpfs. В этом случае монтирование /dev приведет только к созданию /dev/shm как каталога в среде chroot. В этой ситуации мы должны явно смонтировать tmpfs:

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

7.4. Вход в окружение Chroot

Теперь, когда все пакеты, необходимые для сборки остальных инструментов установлены в системе, пришло время войти в окружение chroot и завершить установку временных инструментов. Эта среда также будет использоваться для установки конечной системы. От имени пользователя `root` выполните следующую команду для входа в chroot, в которой на данный момент нет ничего, кроме временных инструментов:

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

Если вы не хотите использовать все доступные логические ядра, замените параметр *`$(nproc)`* количеством логических ядер, которые вы хотите использовать для сборки пакетов в этой и последующих главах. На наборы тестов некоторых пакетов (в частности Autoconf, Libtool и Tar) в Глава 8 не влияет установка переменной `MAKEFLAGS`, вместо этого они используют переменную среды `TESTSUITEFLAGS`. Мы также установили её здесь для запуска тестов с поддержкой нескольких ядер.

Параметр `-i` команды **env**, очистит все переменные в среде chroot. После этого переменные `HOME`, `TERM`, `PS1` и `PATH` будут установлены заново. Конструкция *`TERM=$TERM`* установит переменную TERM внутри chroot в то же значение, что и вне chroot. Эта переменная необходима для корректной работы таких программ как **vim** и **less**. Если понадобятся другие переменные окружения, такие как `CFLAGS` или `CXXFLAGS`, то это подходящее место для их установки.

С этого момента больше нет необходимости использовать переменную `LFS`, поскольку вся работа будет ограничена файловой системой LFS; команда **chroot** запускает оболочку Bash с корневым каталогом (`/`), установленным в `$LFS`.

Обратите внимание, что каталог `/tools/bin` не указан в переменной окружения `PATH`. Это означает, что кросс-тулчейн больше не будет использоваться.

Также обратите внимание, что в командной строке **bash** будет указано `I have no name!`. Это нормально, поскольку файл `/etc/passwd` еще не создан.

Примечание

Важно, чтобы все команды в оставшейся части этой главы и следующих главах выполнялись из среды chroot. Если вы покидаете эту среду по какой-либо причине (например, при перезагрузке), убедитесь, что файловые системы виртуального ядра смонтированы, как описано в Раздел 7.3.1, «Монтирование и заполнение /dev» и Раздел 7.3.2, «Монтирование виртуальных файловых систем ядра», а затем войдите в среду chroot для продолжения установки.

7.5. Создание каталогов

Пришло время создать полную структуру каталогов в файловой системе LFS.

Примечание

Некоторые из каталогов, упомянутых в этом разделе, возможно, уже были созданы ранее с помощью явных инструкций или при установке некоторых пакетов. Они повторяются ниже для полноты картины.

Создайте несколько каталогов, которые не входили в ограниченный набор, используемый в предыдущих главах, выполнив следующую команду:

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

Создайте необходимые подкаталоги, выполнив следующие команды:

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

ln -sfv /run /var/run
ln -sfv /run/lock /var/lock

install -dv -m 0750 /root
install -dv -m 1777 /tmp /var/tmp
```

Необходимые каталоги для поддержки multilib были созданы на предыдущих этапах установки.

По умолчанию каталоги создаются с правами 755, но это нежелательно делать для всех каталогов. В приведенных выше командах вносятся два изменения — одно в домашний каталог пользователя `root`, а другое в каталоги для временных файлов.

Первое изменение гарантирует, что никто не сможет войти в каталог `/root` — точно так же, как обычный пользователь сделал бы это со своим собственным домашним каталогом. Второе изменение гарантирует, что любой пользователь может писать в каталоги `/tmp` и `/var/tmp`, но не может удалять из них файлы другого пользователя. Последнее запрещено так называемым «sticky bit (липким битом)», старшим битом (1) в битовой маске 1777

7.5.1. Примечание о соответствии требованиям FHS

Это дерево каталогов основано на стандарте иерархии файловой системы (FHS) (доступен по адресу *https://refspecs.linuxfoundation.org/fhs.shtml*). FHS также указывает, что наличие некоторых каталогов необязательно, например, `/usr/local/games` и `/usr/share/games`. В LFS мы создаем только те каталоги, которые действительно необходимы. Однако, не стесняйтесь создавать дополнительные каталоги, если хотите.

Предупреждение

FHS не требует наличия каталога `/usr/lib64`, и редакторы LFS решили его не использовать. Чтобы инструкции в LFS и BLFS работали корректно, крайне важно, чтобы этот каталог не существовал. Время от времени вам следует проверять, что он не существует, потому что его легко создать непреднамеренно, и это, вероятно, приведет к поломке вашей системы.

7.6. Создание основных файлов и символических ссылок

Исторически сложилось, что Linux хранит список примонтированных файловых систем в файле `/etc/mtab`. Современные ядра хранят этот список внутри себя и предоставляют его пользователю через файловую систему `/proc`. Чтобы удовлетворять требованиям утилит, которые ожидают наличия `/etc/mtab`, создайте следующую символическую ссылку:

```
ln -sv /proc/self/mounts /etc/mtab
```

Создайте файл `/etc/hosts` , на который будут ссылаться некоторые наборы тестов, а также один из файлов конфигурации Perl:

```
cat > /etc/hosts << EOF
127.0.0.1  localhost $(hostname)
::1        localhost
EOF
```

Чтобы пользователь `root` мог войти в систему и распознавался системой, в файлах `/etc/passwd` и `/etc/group` должны быть соответствующие записи.

Создайте файл `/etc/passwd` выполнив следующую команду:

```
cat > /etc/passwd << "EOF"
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/dev/null:/usr/bin/false
daemon:x:6:6:Daemon User:/dev/null:/usr/bin/false
messagebus:x:18:18:D-Bus Message Daemon User:/run/dbus:/usr/bin/false
uuidd:x:80:80:UUID Generation Daemon User:/dev/null:/usr/bin/false
nobody:x:65534:65534:Unprivileged User:/dev/null:/usr/bin/false
EOF
```

Пароль пользователя `root` будет задан позднее.

Создайте файл `/etc/group` , выполнив следующую команду:

```
cat > /etc/group << "EOF"
root:x:0:
bin:x:1:daemon
sys:x:2:
kmem:x:3:
tape:x:4:
tty:x:5:
daemon:x:6:
floppy:x:7:
disk:x:8:
lp:x:9:
dialout:x:10:
audio:x:11:
video:x:12:
utmp:x:13:
cdrom:x:15:
adm:x:16:
messagebus:x:18:
input:x:24:
mail:x:34:
kvm:x:61:
uuidd:x:80:
wheel:x:97:
users:x:999:
nogroup:x:65534:
EOF
```

Созданные группы не являются частью какого-либо стандарта — это группы, определяемые частично требованиями конфигурации Udev в главе 9, а частично общими соглашениями, используемыми в ряде существующих дистрибутивов Linux. Кроме того, некоторые наборы тестов зависят от конкретных пользователей или групп. Спецификация LSB (доступна по адресу *https://refspecs.linuxfoundation.org/lsb.shtml*) рекомендует, чтобы, помимо группы `root` с идентификатор (GID) 0 присутствовала группа `bin` с GID 1. GID 5 широко используется для группы `tty`, число 5 также используется в `/etc/fstab` для файловой системы `devpts`. Все остальные имена групп и GID могут свободно выбираться системным администратором, так как хорошо написанные программы не зависят от номеров GID, а чаще используют название группы.

Идентификатор 65534 используется ядром для NFS и отдельных пользовательских пространств имен для несопоставленных пользователей и групп (они существуют на сервере NFS или родительском пространстве имен пользователя, но «не существует» на локальном компьютере или в отдельном пространстве имен). Мы присваиваем `nobody` и `nogroup` для того, чтобы избежать несопоставленных идентификаторов. Другие дистрибутивы могут обрабатывать этот идентификатор по-разному, поэтому любая переносимая программа не должна зависеть от этого присвоения.

Для некоторых пакетов требуется локаль.

```
localedef -i C -f UTF-8 C.UTF-8
```

Для некоторых тестов в Глава 8 требуется обычный пользователь. Добавим такого пользователя здесь и удалим эту учетную запись в конце главы.

```
echo "tester:x:101:101::/home/tester:/bin/bash" >> /etc/passwd
echo "tester:x:101:" >> /etc/group
install -o tester -d /home/tester
```

Чтобы удалить приглашение «I have no name!», запустите новую оболочку. Поскольку файлы `/etc/passwd` и `/etc/group` были созданы, разрешение имен пользователей и групп теперь будет работать:

```
exec /usr/bin/bash --login
```

Программы **login**, **agetty**, **init** (и другие) используют ряд журналов для записи такой информации, как кто и когда входил в систему. Однако эти программы не будут записывать данные в журналы, если они еще не существуют. Инициализируйте журналы и предоставьте им соответствующие разрешения:

```
touch /var/log/{btmp,lastlog,faillog,wtmp}
chgrp -v utmp /var/log/lastlog
chmod -v 664  /var/log/lastlog
chmod -v 600  /var/log/btmp
```

В файл `/var/log/wtmp` записываются все входы и выходы из системы. В файл `/var/log/lastlog` записывается время последнего входа каждого пользователя в систему. В файл `/var/log/faillog` записываются неудачные попытки входа в систему. В файл `/var/log/btmp` также записываются неудачные попытки входа в систему.

Примечание

В файл `/run/utmp` записываются пользователи, которые в данный момент вошли в систему. Он создаётся динамически, в процессе выполнения сценариев загрузки.

Примечание

Файлы `utmp`, `wtmp`, `btmp` и `lastlog` используют для временных меток 32-разрядные целые числа, значения счетчика достигнет максимума (2 147 483 647) 19 января 2038 года ("проблема 2038 года"). Многие пакеты перестали их использовать, другие же, собираются прекратить их использование. Вероятно, лучше считать их устаревшими.

7.7. Gettext-0.22.5

Пакет Gettext содержит утилиты для интернационализации и локализации. Они позволяют компилировать программы с поддержкой NLS (Native Language Support), позволяя им выводить сообщения на родном языке пользователя.

Приблизительное время сборки: 1.1 SBU
Требуемое дисковое пространство: 321 MB

7.7.1. Установка пакета Gettext

Для временного набора инструментов нам нужно установить только три программы из пакета Gettext.

Подготовьте Gettext к компиляции:

```
./configure --disable-shared
```

Значение параметров настройки:

`--disable-shared`
В настоящее время нам не нужно устанавливать какие-либо общие библиотеки Gettext, поэтому нет необходимости их собирать.

Скомпилируйте пакет:

```
make
```

Установите программы **msgfmt**, **msgmerge**, и **xgettext** programs:

```
cp -v gettext-tools/src/{msgfmt,msgmerge,xgettext} /usr/bin
```

Подробная информация об этом пакете находится в Раздел 8.34.2, «Содержимое пакета Gettext.»

7.8. Bison-3.8.2

Пакет Bison содержит генератор синтаксического анализа.

Приблизительное время сборки:	0.2 SBU
Требуемое дисковое пространство:	57 MB

7.8.1. Установка пакета Bison

Подготовьте Bison к компиляции:

```
./configure --prefix=/usr \
            --docdir=/usr/share/doc/bison-3.8.2
```

Значение нового параметра конфигурации:

`--docdir=/usr/share/doc/bison-3.8.2`
Этот параметр указывает системе сборки установить документацию к bison в каталог с версией пакета.

Скомпилируйте пакет:

```
make
```

Установите пакет:

```
make install
```

Подробная информация об этом пакете находится в Раздел 8.35.2, «Содержимое пакета Bison.»

7.9. Perl-5.40.0

Пакет Perl содержит практический язык для извлечения данных и составления отчётов (Practical Extraction and Report Language).

Приблизительное время сборки: 0.6 SBU
Требуемое дисковое пространство: 285 MB

7.9.1. Установка пакета Perl

Подготовьте Perl к компиляции:

```
sh Configure -des                                         \
             -D prefix=/usr                               \
             -D vendorprefix=/usr                         \
             -D useshrplib                                \
             -D privlib=/usr/lib/perl5/5.40/core_perl     \
             -D archlib=/usr/lib/perl5/5.40/core_perl     \
             -D sitelib=/usr/lib/perl5/5.40/site_perl     \
             -D sitearch=/usr/lib/perl5/5.40/site_perl    \
             -D vendorlib=/usr/lib/perl5/5.40/vendor_perl \
             -D vendorarch=/usr/lib/perl5/5.40/vendor_perl
```

Значение новых опций Configure:

-des
: Это комбинация из трех параметров: -d использует значения по умолчанию для всех элементов; -e обеспечивает выполнение всех задач; -s отключает несущественные выходные данные.

-D vendorprefix=/usr
: Параметр гарантирует, что **perl** знает, как указать пакетам, где они должны устанавливать свои модули Perl.

-D useshrplib
: Собрать библиотеку libperl, необходимую некоторым модулям Perl, как общую библиотеку вместо статической.

-D privlib, *-D archlib*, *-D sitelib*, *...*
: Эти настройки определяют, где Perl ищет установленные модули. Редакторы LFS решили поместить их в структуру каталогов, основанную на MAJOR.MINOR версии Perl (5.40), что позволяет обновлять Perl до более новых уровней исправлений (уровень исправления - это последняя разделенная точками часть в строке полной версии, например 5.40.0) без необходимости переустанавливать все модули.

Скомпилируйте пакет:

```
make
```

Установите пакет:

```
make install
```

Подробная информация об этом пакете находится в Раздел 8.44.2, «Содержимое пакета Perl.»

7.10. Python-3.12.5

Пакет Python 3 содержит среду разработчика Python. Его можно использовать для объектно-ориентированного программирования, написания скриптов, прототипирования больших программ и разработка целых приложений. Python — это интерпретируемый язык программирования.

Приблизительное время сборки: 0.4 SBU
Требуемое дисковое пространство: 603 MB

7.10.1. Установка пакета Python

Примечание

Существует два пакета, имена которых начинаются с префикса «python». Сейчас необходимо распаковать файл `Python-3.12.5.tar.xz` (обратите внимание на заглавную первую букву).

Подготовка Python к компиляции:

```
./configure --prefix=/usr   \
            --enable-shared \
            --without-ensurepip
```

Значение параметров настройки:

`--enable-shared`
Этот параметр отключает установку статичных библиотек.

`--without-ensurepip`
Этот параметр отключает установщик пакетов Python, который на данном этапе не нужен.

Скомпилируйте пакет:

```
make
```

Примечание

Некоторые модули Python 3 не могут быть собраны сейчас, потому что зависимости еще не установлены. Для модуля `ssl` выводится сообщение `Python требует OpenSSL 1.1.1 или новее` . Сообщение следует проигнорировать. Просто убедитесь, что команда **make** верхнего уровня не завершилась ошибкой. Дополнительные модули сейчас не нужны, и они будут собраны в Глава 8.

Установите пакет:

```
make install
```

Подробная информация об этом пакете находится в Раздел 8.53.2, «Содержимое пакета Python 3.»

7.11. Texinfo-7.1

Пакет Texinfo содержит программы для чтения, записи и преобразования информационных страниц.

Приблизительное время сборки:	0.2 SBU
Требуемое дисковое пространство:	130 MB

7.11.1. Установка пакета Texinfo

Подготовьте Texinfo к компиляции:

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

Скомпилируйте пакет:

```
make
```

Установите пакет:

```
make install
```

Подробная информация об этом пакете находится в Раздел 8.73.2, «Содержимое пакета Texinfo.»

7.12. Util-linux-2.40.2

Пакет Util-linux содержит различные служебные программы. Среди них утилиты для работы с файловыми системами, консолями, разделами и сообщениями.

Приблизительное время сборки: 0.2 SBU
Требуемое дисковое пространство: 180 MB

7.12.1. Установка пакета Util-linux

FHS рекомендует использовать каталог `/var/lib/hwclock` вместо каталога `/etc` в качестве местоположения для файла `adjtime`. Создайте этот каталог:

```
mkdir -pv /var/lib/hwclock
```

Подготовьте Util-linux к компиляции:

```
./configure --libdir=/usr/lib     \
            --runstatedir=/run    \
            --disable-chfn-chsh   \
            --disable-login       \
            --disable-nologin     \
            --disable-su          \
            --disable-setpriv     \
            --disable-runuser     \
            --disable-pylibmount  \
            --disable-static      \
            --disable-liblastlog2 \
            --without-python      \
            ADJTIME_PATH=/var/lib/hwclock/adjtime \
            --docdir=/usr/share/doc/util-linux-2.40.2
```

Значение параметров настройки:

`ADJTIME_PATH=/var/lib/hwclock/adjtime`
Этот параметр устанавливает расположение файла для записи информации об аппаратных часах в соответствии с FHS. Он не обязателен для временного инструментария, но предотвращает создание файла в другом месте, где файл не будет перезаписан или удален при финальной сборке пакета util-linux.

`--libdir=/usr/lib`
Этот параметр гарантирует, что символические ссылки `.so`, будут указывать на файл общей библиотеки в том же каталоге (`/usr/lib`).

`--disable-*`
Этот параметр предотвращают появление предупреждений о сборке компонентов, для которых требуются пакеты, отсутствующие или еще не установленные в LFS.

`--without-python`
Этот параметр отключает использование Python. Это позволяет избежать попыток создания ненужных привязок.

`--runstatedir=/run`
Этот параметр устанавливает расположение сокета, используемого **uuidd** и `libuuid`.

Скомпилируйте пакет:

```
make
```

Установите пакет:

```
make install
```

7.12.2. Установка Util-linux - 32-битная версия

Выполните очистку после предыдущей сборки:

```
make distclean
```

Подготовьте Util-linux к компиляции:

```
CC="gcc -m32" \
./configure --host=i686-pc-linux-gnu \
            --libdir=/usr/lib32      \
            --runstatedir=/run       \
            --docdir=/usr/share/doc/util-linux-2.40.2 \
            --disable-chfn-chsh   \
            --disable-login       \
            --disable-nologin     \
            --disable-su          \
            --disable-setpriv     \
            --disable-runuser     \
            --disable-pylibmount  \
            --disable-static      \
            --disable-liblastlog2 \
            --without-python      \
            ADJTIME_PATH=/var/lib/hwclock/adjtime
```

Скомпилируйте пакет:

```
make
```

Установите пакет:

```
make DESTDIR=$PWD/DESTDIR install
cp -Rv DESTDIR/usr/lib32/* /usr/lib32
rm -rf DESTDIR
```

Подробная информация об этом пакете находится в Раздел 8.80.3, «Содержимое пакета Util-linux.»

7.13. Очистка и сохранение временной системы

7.13.1. Очистка

Во-первых, удалите установленную документацию, чтобы предотвратить ее попадание в конечную систему и сэкономить около 35 МБ места:

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

Во-вторых, в современных системах Linux файлы .la библиотеки libtool полезны только для libltdl. Никакие библиотеки в LFS не загружаются с помощью libltdl. Известно, что некоторые файлы .la могут привести к сбою во время сборки пакетов BLFS. Удалите эти файлы сейчас:

```
find /usr/{lib,libexec} -name \*.la -delete
find /usr/lib32 -name \*.la -delete
```

Сейчас размер системы составляет около 3 ГБ, однако каталог /tools больше не понадобится. Удалите его, чтобы освободить около 1 ГБ дискового пространства:

```
rm -rf /tools
```

7.13.2. Резервное копирование

На данный момент основные программы и библиотеки собраны, и ваша система LFS находится в хорошем состоянии. Можно создать резервную копию вашей системы для последующего повторного использования. В случае фатальных сбоев в следующих главах часто оказывается, что удалить все и начать заново (более осторожно) — лучший вариант восстановления. К сожалению, все временные файлы также будут удалены. Чтобы не тратить лишнее время на повторную сборку того, что было успешно собрано, полезно создать резервную копию текущей системы LFS.

Примечание

Все остальные шаги в этом разделе являются необязательными. Тем не менее, как только вы начнете устанавливать пакеты в Глава 8, временные файлы будут перезаписаны. Поэтому рекомендуется создание резервной копии текущей системы, как описано ниже.

Следующие шаги выполняются вне среды chroot. Это означает, что прежде чем продолжить вы должны покинуть среду chroot. Причиной этого является то, что необходимо получить доступ к расположению файловой системы за пределами среды chroot для хранения/чтения архива резервных копий, который не должен размещаться в иерархии `$LFS`.

Если вы решили сделать резервную копию, покиньте среду chroot:

```
exit
```

Важно

Все следующие инструкции выполняются пользователем `root` в вашей хост-системе. Будьте особенно внимательны к командам, которые вы собираетесь запускать, поскольку ошибки, допущенные здесь, могут изменить вашу хост-систему. Имейте в виду, что переменная окружения `LFS` по умолчанию установлена для пользователя `lfs`, но может не быть установлена для `root`.

Всякий раз, когда команды должны выполняться от `root`, убедитесь, что вы установили переменную `LFS`.

Это обсуждалось в Раздел 2.6, «Установка переменной $LFS».

Перед созданием резервной копии размонтируйте виртуальные файловые системы:

```
mountpoint -q $LFS/dev/shm && umount $LFS/dev/shm
umount $LFS/dev/pts
umount $LFS/{sys,proc,run,dev}
```

Убедитесь, что у вас есть как минимум 1 ГБ свободного места на диске (исходные tar-архивы будут включены в архив резервных копий) в файловой системе, содержащей каталог, в котором вы создаете архив резервных копий.

Обратите внимание, что в приведенных ниже инструкциях указан домашний каталог пользователя `root` хост-системы, который обычно находится в корневой файловой системе. Замените `$HOME` каталогом на ваш выбор, если вы не хотите, чтобы резервная копия хранилась в домашнем каталоге пользователя `root`.

Создайте архив резервной копии, выполнив следующую команду:

Примечание

Поскольку архив резервной копии сжимается, процесс занимает довольно много времени (более 10 минут) даже на достаточно быстрой системе.

```
cd $LFS
tar -cJpf $HOME/lfs-temp-tools-12.2.tar.xz .
```

Примечание

Если вы переходите к главе 8, не забудьте повторно войти в среду chroot, как описано в разделе «Важно» ниже.

7.13.3. Восстановление

В случае, если были допущены какие-либо ошибки и вам нужно начать все сначала, вы можете использовать эту резервную копию для восстановления системы и сэкономить время на восстановление. Поскольку исходники находятся в папке `$LFS`, они также включены в архив резервной копии, поэтому их не нужно загружать повторно. Убедившись, что переменная `$LFS` настроена правильно, вы можете восстановить резервную копию, выполнив следующие команды:

Предупреждение

Следующие команды чрезвычайно опасны. Если вы запустите команду **rm -rf ./*** от имени пользователя `root` и не перейдете в каталог $LFS или переменная окружения `LFS` не будет установлена для пользователя `root`, это уничтожит всю вашу хост-систему. ВЫ ПРЕДУПРЕЖДЕНЫ.

```
cd $LFS
rm -rf ./*
tar -xpf $HOME/lfs-temp-tools-12.2.tar.xz
```

Еще раз проверьте, правильно ли настроено окружение, и продолжайте сборку остальной части системы.

Важно

Если вы покинули среду chroot, чтобы создать резервную копию или перезапустить сборку с помощью восстановления, не забудьте проверить, что виртуальные файловые системы все еще смонтированы (**findmnt | grep $LFS**). Если они не смонтированы, перемонтируйте их сейчас, как описано в Раздел 7.3, «Подготовка виртуальных файловых систем ядра», и повторно войдите в среду chroot (см. Раздел 7.4, «Вход в окружение Chroot»), прежде чем продолжить.

Часть IV. Сборка системы LFS

Глава 8. Установка базового системного программного обеспечения

8.1. Введение

В этой главе мы приступаем к сборке конечной системы LFS.

Установка программного обеспечения проста. Хотя во многих случаях инструкции по установке можно было бы сделать короче и универсальнее, мы решили предоставить полные инструкции для каждого пакета, чтобы свести к минимуму вероятность ошибок. Ключом к пониманию того, что заставляет систему Linux работать, является знание того, для чего используется каждый пакет и зачем он вам (или системе) может понадобиться.

Мы не рекомендуем использовать оптимизации. С ними программа может работать немного быстрее, но также они могут вызвать сложности при компиляции и проблемы при запуске программы. Если пакет не компилируется при использовании оптимизации, попробуйте скомпилировать его без оптимизации и посмотрите, решает ли это проблему. Даже если пакет компилируется при использовании оптимизации, существует риск, что он может быть скомпилирован неправильно из-за сложных взаимодействий между кодом и инструментами сборки. Также обратите внимание, что параметры `-march` и `-mtune`, не тестировались со значениями отличными от указанных в книге. Это может вызвать проблемы с пакетами набора инструментов (Binutils, GCC и Glibc). Небольшие потенциальные плюсы, достигаемые за счет оптимизации, часто перевешиваются рисками. Тем кто собирает LFS впервые рекомендуется делать это без пользовательских оптимизаций.

С другой стороны, мы сохраняем оптимизацию включенной в конфигурации пакетов по умолчанию. Кроме того, иногда мы явно включаем оптимизированную конфигурацию, предоставляемую пакетом, но не включенную по умолчанию. Сопровождающие пакета уже протестировали эти конфигурации и считают их безопасными, поэтому маловероятно, что они сломают сборку. Как правило, конфигурация по умолчанию уже включает параметры `-O2` или `-O3`, поэтому результирующая система по-прежнему будет работать очень быстро без какой-либо пользовательской оптимизации и в то же время будет стабильной.

Перед инструкцией по установке на каждой странице представлена информация о пакете, включая краткое описание того, что он содержит, примерное время, необходимое для сборки, и сколько места на диске требуется в процессе сборки. После инструкции по установке идет список программ и библиотек (вместе с кратким описанием), которые устанавливает пакет.

Примечание

Для всех пакетов в Глава 8 значения SBU и требуемое дисковое пространство указано с учетом тестов. Значения SBU были рассчитаны с использованием четырех ядер ЦП (-j4) для всех операций, если не указано иное.

8.1.1. О библиотеках

Как правило, редакторы LFS не рекомендуют собирать и устанавливать статические библиотеки. Большинство статических библиотек устарели в современной системе Linux. Кроме того, линковка статической библиотеки с программой может быть вредна. Если для устранения проблемы безопасности требуется обновление библиотеки, все программы, использующие статическую библиотеку, необходимо будет повторно перелинковать с новой библиотекой. Поскольку использование статических библиотек не всегда очевидно, соответствующие программы (и процедуры, необходимые для линковки) могут быть даже неизвестны.

В инструкциях этой главы мы удаляем или отключаем установку большинства статических библиотек. Обычно это делается путем передачи параметра `--disable-static` при выполнении **configure**. Иногда необходимо использовать альтернативные методы. В некоторых случаях, в частности в пакетах Glibc и GCC, использование статических библиотек остается важным элементом процесса сборки пакетов.

Более подробное обсуждение библиотек смотрите *Библиотеки: статические или общие?* в книге BLFS.

8.2. Управление пакетами

Управление пакетами — часто спрашиваемое дополнение к книге LFS. Менеджер пакетов позволяет отслеживать установку файлов, упрощая удаление и обновление пакетов. Хороший менеджер пакетов также будет обрабатывать конфигурационные файлы, чтобы сохранить пользовательские настройки при переустановке или обновлении пакета. Прежде чем вы начнете задаваться вопросом, НЕТ — в этом разделе не будет ни говориться, ни рекомендоваться какой-либо конкретный менеджер пакетов. Что он действительно предоставляет, так это обзор наиболее популярных методов и того, как они работают. Идеальным менеджером пакетов для вас может быть один из этих методов или комбинация двух и более методов. В этом разделе кратко упоминаются проблемы, которые могут возникнуть при обновлении пакетов.

Некоторые причины, по которым менеджер пакетов не упоминается в LFS или BLFS представлены ниже:

- Рассмотрение управления пакетами отвлекает внимание от целей этих книг — обучения тому, как строится система Linux.
- Существует множество решений для управления пакетами, каждое из которых имеет свои сильные и слабые стороны. Трудно найти такое, которое удовлетворит всех.

Есть несколько советов, написанных на тему управления пакетами. Посетите проект *Советы* возможно вы найдете решение, которое соответствует вашим потребностям.

8.2.1. Проблемы с обновлением

Менеджер пакетов упрощает обновление до более новых версий после их выпуска. Как правило, инструкции в книгах LFS и BLFS можно использовать для обновления до более новых версий. Вот некоторые моменты, о которых следует помнить при обновлении пакетов, особенно в работающей системе.

- Если нужно обновить ядро Linux (например, с 5.10.17 до 5.10.18 или 5.11.1), дополнительно пересобирать ничего не нужно. Система продолжит нормально работать благодаря четко определенной границе между ядром и пользовательским пространством. В частности, заголовки Linux API не нужно обновлять вместе с ядром. Вам просто нужно перезагрузить систему, чтобы использовать обновленное ядро.
- Если необходимо обновить Glibc до более новой версии (например, с Glibc-2.36 до Glibc-2.40) необходимо выполнить некоторые дополнительные действия, чтобы избежать поломки системы. Подробности читайте в Раздел 8.5, «Glibc-2.40».
- Если пакет, содержащий общую библиотеку, обновляется и имя библиотеки изменилось, то любые пакеты, динамически связанные с библиотекой, необходимо перекомпилировать, чтобы связать с более новой библиотекой. (Обратите внимание, что между версией пакета и именем библиотеки нет никакой связи.) Например, рассмотрим пакет foo-1.2.3, который устанавливает общую библиотеку с именем `libfoo.so.1`. Предположим, вы обновили пакет до более новой версии foo-1.2.4, которая устанавливает общую библиотеку с именем `libfoo.so.2`, все пакеты, которые динамически связаны с `libfoo.so.1`, должны быть перекомпилированы для связи с `libfoo.so.2`, чтобы использовать новую версию библиотеки. Вы не должны удалять старые библиотеки, пока все зависимые пакеты не будут перекомпилированы.
- Если пакет (прямо или косвенно) связан как со старым, так и с новым именем общей библиотеки (например, пакет ссылается как на `libfoo.so.2`, так и на `libbar.so.1`, в то время как последний ссылается на `libfoo.so.3`), пакет может работать неправильно, поскольку разные версии общей

библиотеки содержат несовместимые определения для некоторых имен символов. Это может быть вызвано перекомпиляцией некоторых, но не всех, пакетов, связанных со старой общей библиотекой, после обновления пакета, предоставляющего общую библиотеку. Чтобы избежать этой проблемы, пользователям необходимо как можно скорее пересобрать каждый пакет, связанный с общей библиотекой, с обновленной версией (например, с libfoo.so.2 на libfoo.so.3).

- Если пакет, содержащий общую библиотеку, обновляется, а имя библиотеки не меняется, но уменьшается номер версии **файла** библиотеки (например, библиотека по-прежнему называется `libfoo.so.1`, но имя файла библиотеки изменилось с `libfoo.so.1.25` на `libfoo.so.1.24`), следует удалить файл библиотеки ранее установленной версии (в данном случае `libfoo.so.1.25`). В противном случае, команда **ldconfig** (запущенная самостоятельно с помощью командной строки или при установке какого-либо пакета) приведёт к сбросу символической ссылки `libfoo.so.1`, которая будет указывать на старый файл библиотеки, потому что кажется, что она имеет «более новую» версию, поскольку её номер версии больше. Такая ситуация может произойти, если вам нужно понизить версию пакета или авторы изменили схему управления версиями файлов библиотеки.
- Если пакет, содержащий общую библиотеку, обновляется, а имя библиотеки не меняется, но устраняется серьезная проблема (особенно уязвимость в системе безопасности), необходимо перезапустить все работающие программы, связанные с общей библиотекой. Следующая команда, запущенная от имени пользователя `root` после завершения обновления, выведет список программ, которые использует старые версии этих библиотек (замените *`libfoo`* именем библиотеки):

```
grep -l 'libfoo.*deleted' /proc/*/maps | tr -cd 0-9\\n | xargs -r ps u
```

Если для доступа к системе используется OpenSSH и он связан с обновленной библиотекой, вам необходимо перезапустить службу **sshd**, затем выйти из системы, снова войти в систему и повторно выполнить предыдущую команду, чтобы убедиться, что удаленные библиотеки более не используются.

- Если исполняемая программа или библиотека перезаписаны, процессы, использующие код или данные из них, могут завершиться сбоем. Правильный способ обновить программу или общую библиотеку, не вызывая сбоя процесса, - это сначала удалить его, а затем установить новую версию. Команда **install**, предоставляемая Coreutils, уже реализовала это, и большинство пакетов используют ее для установки двоичных файлов и библиотек. Это означает, что большую часть времени вас не будет беспокоить эта проблема. Однако процесс установки некоторых пакетов (в частности, SpiderMonkey в BLFS) просто перезаписывает файл, если он существует, и вызывает сбой. Поэтому безопаснее сохранить свою работу и закрыть ненужные запущенные программы перед обновлением пакета.

8.2.2. Методы управления пакетами

Ниже приведены некоторые распространенные методы управления пакетами. Прежде чем принять решение о менеджере пакетов, проведите исследование различных методов, особенно недостатки каждой конкретной схемы.

8.2.2.1. Всё у меня в голове!

Да, это метод управления пакетами. Некоторым людям не нужен менеджер пакетов, потому что они хорошо знакомы с пакетами и знают, какие файлы устанавливаются каждым пакетом. Некоторым пользователям также не требуется какое-либо управление пакетами, поскольку они планируют пересобирать всю систему при каждом изменении пакета.

8.2.2.2. Установка в отдельные каталоги

Это упрощенный метод управления пакетами, для которого не требуется специальная программа управления. Каждый пакет устанавливается в отдельный каталог. Например, пакет foo-1.1 устанавливается в `/opt/foo-1.1`, а символическая ссылка создается из `/opt/foo` в `/opt/foo-1.1`. Когда появляется новая версия foo-1.2, она устанавливается в `/opt/foo-1.2` и предыдущая символическая ссылка заменяется символической ссылкой на новую версию.

Переменные окружения, такие как PATH, MANPATH, INFOPATH, PKG_CONFIG_PATH , CPPFLAGS, LDFLAGS и файл конфигурации /etc/ld.so.conf , возможно, потребуется расширить, включив соответствующие подкаталоги в /opt/foo-x.y .

Этот подход используется в книге BLFS для установки некоторых очень больших пакетов, чтобы упростить их обновление. Если вы устанавливаете много таких пакетов, эта схема становится неуправляемой. Некоторые пакеты (например, заголовки Linux API и Glibc) могут плохо работать с такой структурой. **Никогда не используйте её в масштабах всей системы.**

8.2.2.3. Управление пакетами с использованием символических ссылок

Это разновидность предыдущей техники.Каждый пакет устанавливается аналогично, но вместо создания символической ссылки на общее имя пакета, каждому файлу создаётся символическая ссылка в иерархии каталогов /usr . Это исключает необходимость модификации значений переменных окружения. Хотя такие ссылки могут быть созданы пользователем, многие менеджеры пакетов используют именной такой подход. Наиболее популярные из них - Stow, Epkg, Graft и Depot.

Установку нужно сымитировать, чтобы пакет думал, что он установлен в /usr , хотя на самом деле он установлен в иерархии /usr/pkg . Установка таким способом обычно является нетривиальной задачей. Например, предположим, что вы устанавливаете пакет libfoo-1.1. Следующие инструкции могут привести к неправильной установке пакета:

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

Установка будет выполнена, но зависимые пакеты не смогут ссылаться на libfoo. Если вы скомпилируете пакет, который ссылается на libfoo, вы заметите, что он связан с /usr/pkg/libfoo/1.1/lib/libfoo.so.1 вместо /usr/lib/libfoo.so.1 , как вы ожидаете. Правильный подход заключается в использовании переменной DESTDIR для управления установкой. Этот подход работает следующим образом:

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

Большинство пакетов поддерживают этот подход, но есть и такие, которые этого не делают. Для несовместимых пакетов вам может потребоваться либо установить пакет вручную, либо вы можете установить проблемные пакеты в /opt .

8.2.2.4. На основе временной метки

В этом методе файлу присваивается временная метка перед установкой пакета. После установки простое использование команды **find** с соответствующими параметрами может создать журнал всех файлов, установленных после создания файла с временной метки. Менеджером пакетов, использующим этот подход, является install-log.

Хотя преимущество этой схемы в том, что она проста, у нее есть два недостатка. Если во время установки, файлы устанавливаются с отметкой времени, отличной от текущего времени, эти файлы не будут отслеживаться менеджером пакетов. Кроме того, эта схема может использоваться только при установке пакетов по одному. Журналы ненадежны, если два пакета устанавливаются одновременно на двух разных консолях.

8.2.2.5. Отслеживание сценариев установки

При таком подходе, записываются команды, выполняемые сценариями установки. Есть два метода, которые можно использовать:

Переменная среды `LD_PRELOAD` может быть установлена так, чтобы она указывала на библиотеку, которую нужно предварительно загрузить перед установкой. Во время установки эта библиотека отслеживает устанавливаемые пакеты, присоединяясь к различным исполняемым файлам, таким как **cp**, **install**, **mv**, и отслеживая системные вызовы, изменяющие файловую систему. Чтобы этот подход работал, все исполняемые файлы должны быть динамически связаны без битов suid или sgid. Предварительная загрузка библиотеки может вызвать некоторые нежелательные побочные эффекты во время установки. Поэтому рекомендуется выполнить некоторые тесты, чтобы убедиться, что менеджер пакетов ничего не сломает и что он регистрирует все соответствующие файлы.

Другой метод заключается в использовании **strace**, который регистрирует все системные вызовы, сделанные во время выполнения сценариев установки.

8.2.2.6. Создание архивов пакетов

В этой схеме установка пакета имитируется в отдельном дереве, как описано ранее в разделе управление пакетами с использованием символических ссылок. После установки из установленных файлов создается архив пакета. Затем этот архив используется для установки пакета на локальный компьютер или даже на другие компьютеры.

Этот подход используется большинством менеджеров пакетов, имеющихся в коммерческих дистрибутивах. Примерами менеджеров пакетов, которые следуют этому подходу, являются RPM (который, кстати, требуется согласно спецификации *Linux Standard Base Specification*), pkg-utils, apt Debian и система Portage Gentoo. Описание того, как использовать этот стиль управления пакетами для систем LFS, находится по адресу *https://mirror.linuxfromscratch.ru/hints/downloads/files/fakeroot.txt*.

Создание файлов пакетов, содержащих информацию о зависимостях, является сложной задачей и выходит за рамки LFS.

Slackware использует систему на основе **tar** для архивов пакетов. Эта система намеренно не обрабатывает зависимости пакетов, как это делают более сложные менеджеры пакетов. Подробнее об управлении пакетами Slackware см. *https://www.slackbook.org/html/package-management.html*.

8.2.2.7. Пользовательское управление пакетами

Эта схема, уникальная для LFS, была разработана Маттиасом Бенкманом и доступна в проекте *Hints*. В этой схеме каждый пакет устанавливается отдельным пользователем в стандартные папки. Файлы, принадлежащие пакету, легко идентифицируются путем проверки идентификатора пользователя. Особенности и недостатки этого подхода слишком сложны, чтобы описывать их в этом разделе. Для получения более подробной информации, пожалуйста, ознакомьтесь с советами по адресу *https://mirror.linuxfromscratch.ru/hints/downloads/files/more_control_and_pkg_man.txt*.

8.2.3. Развертывание LFS на нескольких системах

Одним из преимуществ системы LFS является отсутствие файлов, зависящих от положения файлов на диске. Клонировать сборку LFS на другой компьютер с той же архитектурой, что и у базовой системы, так же просто, как использовать **tar** для архивации раздела LFS, содержащем корневой каталог (около 900 МБ в несжатом виде для базовой сборки LFS), скопировать этот файл по сети или с помощью CD / USB носителя в новую систему и распаковать его. После этого необходимо изменить несколько конфигурационных файлов. Файлы, которые, возможно, потребуется изменить представлены в списке ниже: `/etc/hosts`, `/etc/fstab`, `/etc/passwd`, `/etc/group`, `/etc/shadow`, `/etc/ld.so.conf`, `/etc/sysconfig/rc.site`, `/etc/sysconfig/network`, и `/etc/sysconfig/ifconfig.eth0`.

Возможно, потребуется собрать собственное ядро для новой системы в зависимости от различий в системном оборудовании и исходной конфигурации ядра.

Примечание

Поступали сообщения о некоторых проблемах при копировании между похожими, но не идентичными архитектурами. Например, набор инструкций для Intel не идентичен набору инструкций для процессора AMD, и более поздние версии некоторых процессоров могут содержать инструкции, недоступные в более ранних версиях.

Наконец, новую систему необходимо сделать загрузочной так, как это описано в Раздел 10.4, «Использование GRUB для настройки процесса загрузки».

8.3. Man-pages-6.9.1

Пакет Man-pages содержит более 2400 справочных руководств.

Приблизительное время сборки: 0.1 SBU
Требуемое дисковое пространство: 52 MB

8.3.1. Установка пакета Man-pages

Удалите две справочные страницы для функций хэширования паролей. Libxcrypt предоставит улучшенную версию этих справочных страниц:

```
rm -v man3/crypt*
```

Установите пакет Man-pages выполнив команду:

```
make prefix=/usr install
```

8.3.2. Содержимое пакета Man-pages

Установленные файлы: различные справочные страницы

Краткое описание

`man pages` Описывают функции языка программирования C, важные файлы устройств и важные файлы конфигурации.

8.4. Iana-Etc-20240806

Пакет Iana-Etc предоставляет данные для сетевых служб и протоколов.

Приблизительное время сборки: менее 0.1 SBU
Требуемое дисковое пространство: 4.8 MB

8.4.1. Установка пакета Iana-Etc

Для этого пакета необходимо лишь скопировать нужные файлы:

```
cp services protocols /etc
```

8.4.2. Содержимое пакета Iana-Etc

Установленные файлы: /etc/protocols и /etc/services

Краткое описание

`/etc/protocols` Описывает различные интернет-протоколы DARPA, которые доступны из подсистемы TCP/IP

`/etc/services` Обеспечивает сопоставление понятных текстовых имен для интернет-сервисов с назначенными им номерами портов и типами протоколов.

8.5. Glibc-2.40

Пакет Glibc содержит основную библиотеку C. Эта библиотека предоставляет основные процедуры для выделения памяти, поиска в каталогах, открытия и закрытия файлов, чтения и записи файлов, обработки строк, сопоставления с образцом, арифметики и так далее

Приблизительное время сборки: 12 SBU
Требуемое дисковое пространство: 3.1 GB

8.5.1. Установка пакета Glibc

Некоторые программы Glibc используют не совместимый с FHS каталог `/var/db` для хранения своих данных во время выполнения. Примените следующий патч, чтобы эти программы хранили свои данные в каталогах, совместимых с FHS:

```
patch -Np1 -i ../glibc-2.40-fhs-1.patch
```

Документация Glibc рекомендует выполнять компиляцию в отдельном каталоге:

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

Убедитесь, что утилиты **ldconfig** и **sln** будут установлены в `/usr/sbin` :

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

Подготовьте Glibc к компиляции:

```
../configure --prefix=/usr                            \
             --disable-werror                         \
             --enable-kernel=4.19                     \
             --enable-stack-protector=strong          \
             --disable-nscd                           \
             libc_cv_slibdir=/usr/lib
```

Значение параметров настройки:

`--disable-werror`
Отключает параметр -Werror, передаваемый GCC. Это необходимо для запуска набора тестов.

`--enable-kernel=4.19`
Этот параметр сообщает системе сборки, что Glibc может использоваться с ядрами старше 4.19. Это значение используется для создание обходных путей на случай, если системный вызов, представленный в более поздней версии, нельзя будет использовать.

`--enable-stack-protector=strong`
Этот параметр повышает безопасность системы за счет добавления дополнительного кода для проверки переполнения буфера. Обратите внимание, что Glibc всегда явно переопределяет параметры GCC по умолчанию, поэтому необходимо всегда указывать эту опцию, несмотря на то, что мы уже указали `--enable-default-ssp` для GCC.

`--disable-nscd`
Параметр отключает сборку демона кэша службы имен, который больше не используется.

`libc_cv_slibdir=/usr/lib`
Эта переменная устанавливает правильную библиотеку для всей системы. Мы не хотим, чтобы использовалась lib64

Скомпилируйте пакет:

```
make
```

Важно

В этом разделе набор тестов для Glibc считается критически важным. Ни в коем случае не пропускайте его.

Как правило, несколько тестов не проходят. Ошибки тестирования, перечисленные ниже, можно игнорировать.

```
make check
```

Вы можете увидеть, что ряд тестов завершились неудачей. Набор тестов Glibc в некоторой степени зависит от хост-системы. Несколько ошибок из более чем 5000 тестов можно игнорировать. Список наиболее распространенных проблем последних версий LFS:

- Известно, что *io/tst-lchmod* не работает в среде chroot LFS.
- Известно, что некоторые тесты, например *nss/tst-nss-files-hosts-multi* и *nptl/tst-thread-affinity** завершаются неудачей из-за тайм-аута (особенно когда система работает относительно медленно и/или набор тестов запущен в несколько потоков). Эти тесты могут быть идентифицированы с помощью следующей команды:

  ```
  grep "Timed out" $(find -name \*.out)
  ```

  Можно повторно запустить отдельный тест, увеличив таймаут с помощью команды **TIMEOUTFACTOR=*<factor>* make test t=*<test name>***. Например, **TIMEOUTFACTOR=10 make test t=nss/tst-nss-files-hosts-multi** перезапустит *nss/tst-nss-files-hosts-multi*, увеличив начальный таймаут в 10 раз.
- Кроме того, некоторые тесты могут завершиться неудачно при использовании относительно старой модели процессора (например, *elf/tst-cpu-features-cpuinfo*) или версии ядра хоста (например, *stdlib/tst-arc4random-thread*).

На этапе установки Glibc будет жаловаться на отсутствие файла `/etc/ld.so.conf` , хотя это безобидное сообщение, предотвратить его появление можно с помощью команды:

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

Исправьте Makefile, чтобы пропустить устаревшую проверку работоспособности, которая завершается неудачей в современной конфигурации Glibc:

```
sed '/test-installation/s@$(PERL)@echo not running@' -i ../Makefile
```

Важно

При обновлении Glibc до новой минорной версии (например, с Glibc-2.36 до Glibc-2.40) в работающей системе LFS вам необходимо принять некоторые дополнительные меры предосторожности, чтобы избежать поломки системы:

- Обновление Glibc в системе LFS до версии 11.0 не поддерживается. Пересоберите LFS, если вы используете такую старую систему, но вам нужна более новая Glibc.
- При обновлении системы LFS до версии 12.0 установите Libxcrypt следуя инструкции Раздел 8.28, «Libxcrypt-4.4.36.». В дополнение к обычной установке Libxcrypt, **вы ДОЛЖНЫ следовать примечанию со страницы Libxcrypt, чтобы установить `libcrypt.so.1*` (заменив `libcrypt.so.1` из предыдущей установки Glibc)**.
- При обновлении системы LFS до версии 12.1 удалите программу **nscd**:

```
rm -f /usr/sbin/nscd
```

- Обновите ядро и перезагрузитесь, если оно старше 4.19 (проверьте текущую версию с помощью **uname -r**) или, если вы хотите обновить имеющееся ядро, выполните действия из Раздел 10.3, «Linux-6.10.5.»
- Обновите заголовочные файлы API ядра, если они старше 4.19 (проверьте текущую версию с помощью **cat /usr/include/linux/version.h**) или, если вы просто хотите обновить их, следуйте Раздел 5.4, «Заголовочные файлы Linux-6.10.5 API» (но удалив `$LFS` из команды **cp**).
- Выполните установку `DESTDIR` и обновите общие библиотеки Glibc в системе с помощью одной команды **install**:

```
make DESTDIR=$PWD/dest install
install -vm755 dest/usr/lib/*.so.* /usr/lib
```

Крайне важно строго следовать описанным выше шагам, если вы не совсем понимаете, что делаете. **Любое неожиданное отклонение может сделать систему полностью непригодной для использования. ВЫ ПРЕДУПРЕЖДЕНЫ.**

Затем продолжайте выполнять команды **make install**, **sed** для `/usr/bin/ldd` и команды для установки локалей. Как только они будут выполнены, перезагрузите систему.

Установите пакет:

```
make install
```

Исправьте жестко заданный путь к исполняемому загрузчику в скрипте **ldd**:

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

Затем установите локали, которые дадут возможность системе отвечать на разных языках. Ни одна из локалей не требуется системе, но если некоторые из них отсутствуют, то наборы тестов ряда пакетов будут пропускать важные тестовые сценарии.

Отдельные локали можно установить с помощью программы **localedef**. Например, вторая команда **localedef** приведенная ниже, объединяет определение независимой от набора символов локали `/usr/share/i18n/locales/cs_CZ` с набором символов `/usr/share/i18n/charmaps/UTF-8.gz` и добавляет результат в файл `/usr/lib/locale/locale-archive`. Следующие инструкции установят минимальный набор локалей, необходимый для оптимального охвата тестов

```
localedef -i C -f UTF-8 C.UTF-8
localedef -i cs_CZ -f UTF-8 cs_CZ.UTF-8
localedef -i de_DE -f ISO-8859-1 de_DE
localedef -i de_DE@euro -f ISO-8859-15 de_DE@euro
localedef -i de_DE -f UTF-8 de_DE.UTF-8
localedef -i el_GR -f ISO-8859-7 el_GR
localedef -i en_GB -f ISO-8859-1 en_GB
localedef -i en_GB -f UTF-8 en_GB.UTF-8
localedef -i en_HK -f ISO-8859-1 en_HK
localedef -i en_PH -f ISO-8859-1 en_PH
localedef -i en_US -f ISO-8859-1 en_US
localedef -i en_US -f UTF-8 en_US.UTF-8
localedef -i es_ES -f ISO-8859-15 es_ES@euro
localedef -i es_MX -f ISO-8859-1 es_MX
localedef -i fa_IR -f UTF-8 fa_IR
localedef -i fr_FR -f ISO-8859-1 fr_FR
localedef -i fr_FR@euro -f ISO-8859-15 fr_FR@euro
localedef -i fr_FR -f UTF-8 fr_FR.UTF-8
localedef -i is_IS -f ISO-8859-1 is_IS
localedef -i is_IS -f UTF-8 is_IS.UTF-8
localedef -i it_IT -f ISO-8859-1 it_IT
localedef -i it_IT -f ISO-8859-15 it_IT@euro
localedef -i it_IT -f UTF-8 it_IT.UTF-8
localedef -i ja_JP -f EUC-JP ja_JP
localedef -i ja_JP -f SHIFT_JIS ja_JP.SJIS 2> /dev/null || true
localedef -i ja_JP -f UTF-8 ja_JP.UTF-8
localedef -i nl_NL@euro -f ISO-8859-15 nl_NL@euro
localedef -i ru_RU -f KOI8-R ru_RU.KOI8-R
localedef -i ru_RU -f UTF-8 ru_RU.UTF-8
localedef -i se_NO -f UTF-8 se_NO.UTF-8
localedef -i ta_IN -f UTF-8 ta_IN.UTF-8
localedef -i tr_TR -f UTF-8 tr_TR.UTF-8
localedef -i zh_CN -f GB18030 zh_CN.GB18030
localedef -i zh_HK -f BIG5-HKSCS zh_HK.BIG5-HKSCS
localedef -i zh_TW -f UTF-8 zh_TW.UTF-8
```

Кроме того, установите локаль для вашей страны, языка и набора символов.

В качестве альтернативы, установите сразу все локали перечисленные в файле `glibc-2.40/localedata/SUPPORTED` (он включает все локали из списка выше и многие другие), выполнив команду:

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

Затем используйте команду **localedef** для создания и установки локалей, не перечисленных в файле `glibc-2.40/localedata/SUPPORTED`, когда они вам понадобятся. Например, для некоторых тестов в этой главе потребуются следующие две локали:

```
localedef -i C -f UTF-8 C.UTF-8
localedef -i ja_JP -f SHIFT_JIS ja_JP.SJIS 2> /dev/null || true
```

Примечание

Glibc теперь использует libidn2 для разрешения интернационализированных доменных имен. Если такая функция необходима, то инструкцию по установке libidn2 можно найти на странице *BLFS libidn2*.

8.5.2. Настройка Glibc

8.5.2.1. Добавление nsswitch.conf

Необходимо создать файл `/etc/nsswitch.conf` , потому что настроенный по умолчанию Glibc плохо работает в сетевой среде.

Создайте новый фай `/etc/nsswitch.conf` , выполнив следующие действия:

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

passwd: files
group: files
shadow: files

hosts: files dns
networks: files

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

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

8.5.2.2. Добавление данных о часовом поясе

Установите и настройте часовой пояс следующим образом:

```
tar -xf ../../tzdata2024a.tar.gz

ZONEINFO=/usr/share/zoneinfo
mkdir -pv $ZONEINFO/{posix,right}

for tz in etcetera southamerica northamerica europe africa antarctica  \
          asia australasia backward; do
    zic -L /dev/null   -d $ZONEINFO       ${tz}
    zic -L /dev/null   -d $ZONEINFO/posix ${tz}
    zic -L leapseconds -d $ZONEINFO/right ${tz}
done

cp -v zone.tab zone1970.tab iso3166.tab $ZONEINFO
zic -d $ZONEINFO -p America/New_York
unset ZONEINFO
```

Значение команд zic:

`zic -L /dev/null ...`
: Создаёт часовые пояса posix без секунд координации. Обычно их помещают как в `zoneinfo` так и в `zoneinfo/posix` . Часовые пояса POSIX должны быть прописаны в `zoneinfo`, иначе различные тесты будут сообщать об ошибках. На встраиваемых системах с небольшим диском, где часовые пояса никогда не будут обновляться, можно сэкономить примерно 1.9 MB не используя каталог `posix`, однако некоторые приложения или наборы тестов могут вызывать сбои.

`zic -L leapseconds ...`
: Создаёт правильные часовые пояса с секундами координации. На встраиваемых системах с небольшим диском, где часовые пояса никогда не будут обновляться, а правильность времени неважна, можно выиграть примерно 1.9 MB, исключив каталог `right`.

`zic ... -p ...`
: Создаёт файл `posixrules`. Используется New York, потому что POSIX требует, чтобы правила перехода на летнее время соответствовали правилам США.

Один из способов определить местный часовой пояс — запустить следующий скрипт:

```
tzselect
```

После нескольких вопросов о местоположении скрипт выдаст наименование часового пояса (например *America/Edmonton*). В файле `/usr/share/zoneinfo` перечислены и другие возможные часовые пояса, такие как *Canada/Eastern* или *EST5EDT*, которые не распознаются скриптом, но могут быть использованы.

Создайте файл `/etc/localtime` выполнив:

```
ln -sfv /usr/share/zoneinfo/<xxx> /etc/localtime
```

Замените *<xxx>* на имя выбранного часового пояса (например, Europe/Moscow).

8.5.2.3. Настройка динамического загрузчика

По умолчанию, динамический загрузчик (`/lib/ld-linux.so.2`) ищет в каталоге `/usr/lib` , нужные для работы программ библиотеки. Однако, если библиотеки находятся в другом каталоге, то его необходимо указать в файле `/etc/ld.so.conf` , чтобы динамический загрузчик мог их найти. Два каталога - `/usr/local/lib` и `/opt/lib` часто используются для дополнительных библиотек, поэтому добавьте их в пути поиска для динамического загрузчика.

Создайте новый файл `/etc/ld.so.conf` выполнив:

```
cat > /etc/ld.so.conf << "EOF"
# Begin /etc/ld.so.conf
/usr/local/lib
/opt/lib

EOF
```

Динамический загрузчик может выполнить поиск в каталоге и включить содержимое найденных там файлов. Обычно такие файлы состоят из одной строки и содержат путь к библиотеке. Чтобы добавить эту возможность, выполните следующие команды:

```
cat >> /etc/ld.so.conf << "EOF"
# Add an include directory
include /etc/ld.so.conf.d/*.conf

EOF
mkdir -pv /etc/ld.so.conf.d
```

8.5.3. Установка Glibc — 32-битная версия

Теперь перекомпилируйте Glibc для m32. Извлеченный исходный код можно использовать повторно, но его необходимо очистить перед установкой Glibc m32.

Очистите каталог сборки и удалите артефакты от предыдущей сборки:

```
rm -rf ./*
find .. -name "*.a" -delete
```

Сконфигурируйте Glibc для m32 с помощью следующих команд:

```
CC="gcc -m32" CXX="g++ -m32" \
../configure                             \
      --prefix=/usr                      \
      --host=i686-pc-linux-gnu           \
      --build=$(../scripts/config.guess) \
      --enable-kernel=4.19                \
      --disable-nscd                     \
      --libdir=/usr/lib32                \
      --libexecdir=/usr/lib32            \
      libc_cv_slibdir=/usr/lib32
```

Скомпилируйте пакет:

```
make
```

Установите пакет:

```
make DESTDIR=$PWD/DESTDIR install
cp -a DESTDIR/usr/lib32/* /usr/lib32/
install -vm644 DESTDIR/usr/include/gnu/{lib-names,stubs}-32.h \
               /usr/include/gnu/
```

Добавьте имя библиотеки в файл конфигурации динамического загрузчика:

```
echo "/usr/lib32" >> /etc/ld.so.conf
```

Внимание

На этом этапе необходимо остановиться и убедиться, что основные функции (компиляция и компоновка) нового набора инструментов работают должным образом. Чтобы выполнить проверку работоспособности, выполните следующие команды:

```
echo 'int main(){}' > dummy.c
gcc -m32 dummy.c
readelf -l a.out | grep '/ld-linux'
```

Если все работает правильно, ошибок быть не должно, а вывод последней команды будет иметь вид:

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

Если выходные данные отображаются не так, как указано выше, или их вообще нет, значит, что-то сделано неправильно. Разберитесь с проблемой и повторите шаги выше, чтобы исправить ее. Эта проблема должна быть решена, прежде чем вы продолжите.

Как только все будет хорошо, удалите тестовые файлы:

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

8.5.4. Содержимое пакета Glibc

Установленные программы:	gencat, getconf, getent, iconv, iconvconfig, ldconfig, ldd, lddlibc4, ld.so (symlink to ld-linux-x86-64.so.2 or ld-linux.so.2), locale, localedef, makedb, mtrace, pcprofiledump, pldd, sln, sotruss, sprof, tzselect, xtrace, zdump и zic
Установленные библиотеки:	ld-linux-x86-64.so.2, ld-linux.so.2, libBrokenLocale.{a,so}, libanl.{a,so}, libc.{a,so}, libc_nonshared.a, libc_malloc_debug.so, libdl.{a,so.2}, libg.a, libm.{a,so}, libmcheck.a, libmemusage.so, libmvec.{a,so}, libnsl.so.1, libnss_compat.so, libnss_dns.so, libnss_files.so, libnss_hesiod.so, libpcprofile.so, libpthread.{a,so.0}, libresolv.{a,so}, librt.{a,so.1}, libthread_db.so и libutil.{a,so.1}
Созданные каталоги:	/usr/include/arpa, /usr/include/bits, /usr/include/gnu, /usr/include/net, /usr/include/netash, /usr/include/netatalk, /usr/include/netax25, /usr/include/neteconet, /usr/include/netinet, /usr/include/netipx, /usr/include/netiucv, /usr/include/netpacket, /usr/include/netrom, /usr/include/netrose, /usr/include/nfs, /usr/include/protocols, /usr/include/rpc, /usr/include/sys, /usr/lib/audit, /usr/lib/gconv, /usr/lib/locale, /usr/libexec/getconf, /usr/share/i18n, /usr/share/zoneinfo и /var/lib/nss_db

Краткое описание

gencat	Создает каталоги сообщений

getconf	Отображает настройки системы для специфичных переменных файловой системы
getent	Получает записи из административной базы данных
iconv	Выполняет преобразование набора символов
iconvconfig	Создает быстрозагружаемые файлы настроек модуля **iconv**
ldconfig	Создаёт необходимые привязки и кэш динамических библиотек, который используется компоновщиком для связывания во время выполнения
ldd	Сообщает, какие общие библиотеки требуются каждой программе или общей библиотеке
lddlibc4	Помогает **ldd** работать с объектными файлами. Он не существует на более новых архитектурах, таких как x86_64
locale	Выводит различную информацию о текущей локали
localedef	Компилирует спецификации локали
makedb	Создает простую базу данных на основе текстового ввода
mtrace	Читает и интерпретирует файл трассировки памяти; отображает сводку в удобочитаемом формате
pcprofiledump	Создает дамп информации, генерируемой при профилировании ПК
pldd	Перечисляет динамические общие объекты, используемые запущенными процессами.
sln	Статически скомпонованная программа **ln**
sotruss	Отслеживает вызовы процедур общей библиотеки указанной команды
sprof	Читает и отображает данные профилирования общих объектов.
tzselect	Запрашивает у пользователя информацию о текущем местоположении системы и выводит описание соответствующего часового пояса.
xtrace	Отслеживает выполнение программы, отображая выполняемую в данный момент функцию
zdump	Выдает дамп часового пояса
zic	Компилятор часовых поясов
`ld-*.so`	Вспомогательная программа для исполняемых файлов общей библиотеки
`libBrokenLocale`	Используется внутри Glibc как грубый хак для запуска сломанных программ (например, некоторые приложения Motif). Прочитайте комментарии в `glibc-2.40/locale/broken_cur_max.c` для получения дополнительной информации
`libanl`	Библиотека-заглушка, не содержащая функций. Ранее это была библиотека асинхронного поиска имен, функции которой теперь находятся в `libc`
`libc`	Основная библиотека C
`libc_malloc_debug`	Включает проверку выделения памяти при предварительной загрузке
`libdl`	Библиотека-заглушка, не содержащая функций. Ранее была библиотекой интерфейса динамической компоновки, функции которой теперь находятся в `libc`
`libg`	Библиотека-заглушка без функций. Раньше была библиотекой среды выполнения для **g++**
`libm`	Математическая библиотека

`libmvec`	Библиотека векторных математических вычислений, подключаемая по мере необходимости при использовании `libm`
`libmcheck`	Включает проверку выделения памяти при подключении к
`libmemusage`	Используется **memusage** для сбора информации об использовании памяти программой
`libnsl`	Библиотека сетевых служб, которая в настоящее время устарела
`libnss_*`	Модули Name Service Switch, содержащие функции для разрешения имен хостов, имен пользователей, имен групп, псевдонимов, служб, протоколов и т. д. Загружаются `libc` в соответствии с конфигурацией в `/etc/nsswitch.conf`
`libpcprofile`	Содержит функции профилирования, используемые для отслеживания времени, потраченного процессором в конкретных строках исходного кода
`libpthread`	Библиотека-заглушка, не содержащая функций. Ранее содержала функции, обеспечивающие большинство интерфейсов, заданных POSIX.1c Threads Extensions (расширения реализации потоков) и интерфейсы семафоров, указанных в POSIX.1b Real-time Extension (расширения реального времени), теперь эти функции находятся в `libc`
`libresolv`	Содержит функции создания, пересылки и интерпретации пакетов, используемых на серверах доменных имен в сети интернет
`librt`	Содержит функции, реализующие большую часть интерфейсов, определяемых в POSIX.1b Real-time Extension (расширения реального времени)
`libthread_db`	Содержит функции, полезные для сборки отладчиков для многопоточных программ
`libutil`	Библиотека-заглушка, не содержащая функций. Ранее содержал код для «стандартных» функций, используемых во многих утилитах Unix. Эти функции теперь находятся в `libc`

8.6. Zlib-1.3.1

Пакет Zlib содержит подпрограммы сжатия и распаковки, используемые некоторыми программами.

Приблизительное время сборки: менее 0.1 SBU
Требуемое дисковое пространство: 6.4 MB

8.6.1. Установка пакета Zlib

Подготовьте Zlib к компиляции:

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

Скомпилируйте пакет:

```
make
```

Чтобы протестировать пакет, выполните:

```
make check
```

Установите пакет:

```
make install
```

Удалите бесполезную статическую библиотеку:

```
rm -fv /usr/lib/libz.a
```

8.6.2. Установка Zlib - 32-битная версия

Выполните очистку после предыдущей сборки:

```
make distclean
```

Подготовьте Zlib к компиляции:

```
CFLAGS+=" -m32" CXXFLAGS+=" -m32" \
./configure --prefix=/usr \
    --libdir=/usr/lib32
```

Скомпилируйте пакет:

```
make
```

Установите пакет:

```
make DESTDIR=$PWD/DESTDIR install
cp -Rv DESTDIR/usr/lib32/* /usr/lib32
rm -rf DESTDIR
```

8.6.3. Содержимое пакета Zlib

Установленные библиотеки: libz.so

Краткое описание

`libz` Содержит функции сжатия и распаковки, используемые некоторыми программами.

8.7. Bzip2-1.0.8

Пакет Bzip2 содержит программы для сжатия и распаковки файлов. Сжатие текстовых файлов с помощью **bzip2** даёт больший процент сжатия, чем традиционный **gzip**.

Приблизительное время сборки: менее 0.1 SBU
Требуемое дисковое пространство: 7.2 MB

8.7.1. Установка пакета Bzip2

Примените патч, который установит документацию для этого пакета:

```
patch -Np1 -i ../bzip2-1.0.8-install_docs-1.patch
```

Следующая команда гарантирует установку символических ссылок с относительным путём:

```
sed -i 's@\(ln -s -f \)$(PREFIX)/bin/@\1@' Makefile
```

Убедитесь, что справочные страницы установлены в правильном месте:

```
sed -i "s@(PREFIX)/man@(PREFIX)/share/man@g" Makefile
```

Подготовьте Bzip2 к компиляции:

```
make -f Makefile-libbz2_so
make clean
```

Значение параметра make:

-f Makefile-libbz2_so
Этот параметр позволяет выполнить сборку, с использованием другого `Makefile`, в данном случае `Makefile-libbz2_so` , который создает динамическую библиотеку `libbz2.so` и связывает с ней Bzip2.

Скомпилируйте и протестируйте пакет:

```
make
```

Установите пакет:

```
make PREFIX=/usr install
```

Установите библиотеку:

```
cp -av libbz2.so.* /usr/lib
ln -sv libbz2.so.1.0.8 /usr/lib/libbz2.so
```

Установите общий бинарный файл **bzip2** в каталог `/usr/bin` , и замените две копии **bzip2** символическими ссылками:

```
cp -v bzip2-shared /usr/bin/bzip2
for i in /usr/bin/{bzcat,bunzip2}; do
  ln -sfv bzip2 $i
done
```

Удалите ненужную статическую библиотеку:

```
rm -fv /usr/lib/libbz2.a
```

8.7.2. Установка Bzip2 - 32-битная версия

Выполните очистку после предыдущей сборки:

```
make clean
```

Скомпилируйте пакет:

```
sed -e "s/^CC=.*/CC=gcc -m32/" -i Makefile{,-libbz2_so}
make -f Makefile-libbz2_so
make libbz2.a
```

Установите пакет:

```
install -Dm755 libbz2.so.1.0.8 /usr/lib32/libbz2.so.1.0.8
ln -sf libbz2.so.1.0.8 /usr/lib32/libbz2.so
ln -sf libbz2.so.1.0.8 /usr/lib32/libbz2.so.1
ln -sf libbz2.so.1.0.8 /usr/lib32/libbz2.so.1.0
install -Dm644 libbz2.a /usr/lib32/libbz2.a
```

8.7.3. Содержимое пакета Bzip2

Установленные программы:	bunzip2 (ссылка на bzip2), bzcat (ссылка на bzip2), bzcmp (ссылка на bzdiff), bzdiff, bzegrep (ссылка на bzgrep), bzfgrep (ссылка на bzgrep), bzgrep, bzip2, bzip2recover, bzless (ссылка на bzmore) и bzmore
Установленные библиотеки:	libbz2.so
Созданные каталоги:	/usr/share/doc/bzip2-1.0.8

Краткое описание

bunzip2	Распаковывает bzip-файлы
bzcat	Распаковывает в поток стандартного вывода
bzcmp	Запускает программу **cmp** для bzip файлов
bzdiff	Запускает программу **diff** для bzip файлов
bzegrep	Запускает программу **egrep** для bzip файлов
bzfgrep	Запускает программу **fgrep** для bzip файлов
bzgrep	Запускает программу **grep** для bzip файлов
bzip2	Сжимает файлы, используя алгоритм сжатия текста с блочной сортировкой Барроуза — Уилера и кодирование Хафмана; степень сжатия лучше, чем у более традиционных архиваторов, использующих алгоритмы «Lempel-Ziv», например **gzip**
bzip2recover	Пытается восстанавливать данные из поврежденных архивов
bzless	Запускает программу **less** для bzip файлов
bzmore	Запускает программу **more** для bzip файлов
`libbz2`	Библиотека, реализующая сжатие данных без потерь с использованием алгоритма Барроуза-Уилера.

8.8. Xz-5.6.2

Пакет Xz содержит программы для сжатия и распаковки файлов. Он предоставляет возможности для lzma и более новых форматов сжатия xz. Сжатие текстовых файлов с помощью **xz** дает лучший процент сжатия, чем с традиционные **gzip** или **bzip2**.

Приблизительное время сборки: 0.1 SBU

Требуемое дисковое пространство: 21 MB

8.8.1. Установка пакета Xz

Подготовьте Xz к компиляции:

```
./configure --prefix=/usr    \
            --disable-static \
            --docdir=/usr/share/doc/xz-5.6.2
```

Скомпилируйте пакет:

```
make
```

Чтобы протестировать пакет, выполните:

```
make check
```

Установите пакет:

```
make install
```

8.8.2. Установка Xz - 32-битная версия

Выполните очистку после предыдущей сборки:

```
make distclean
```

Подготовьте Xz к компиляции:

```
CC="gcc -m32" ./configure \
    --host=i686-pc-linux-gnu      \
    --prefix=/usr                 \
    --libdir=/usr/lib32           \
    --disable-static
```

Скомпилируйте пакет:

```
make
```

Установите пакет:

```
make DESTDIR=$PWD/DESTDIR install
cp -Rv DESTDIR/usr/lib32/* /usr/lib32
rm -rf DESTDIR
```

8.8.3. Содержимое пакета Xz

Установленные программы:	lzcat (ссылка на xz), lzcmp (ссылка на xzdiff), lzdiff (ссылка на xzdiff), lzegrep (ссылка на xzgrep), lzfgrep (ссылка на xzgrep), lzgrep (ссылка на xzgrep), lzless (ссылка на xzless), lzma (ссылка на xz), lzmadec, lzmainfo, lzmore (ссылка на xzmore), unlzma (ссылка на xz), unxz (ссылка на xz), xz, xzcat (ссылка на xz), xzcmp (ссылка на xzdiff), xzdec, xzdiff, xzegrep (ссылка на xzgrep), xzfgrep (ссылка на xzgrep), xzgrep, xzless и xzmore
Установленные библиотеки:	liblzma.so
Созданные каталоги:	/usr/include/lzma и /usr/share/doc/xz-5.6.2

Краткое описание

lzcat	Распаковывает в стандартный поток вывода
lzcmp	Запускает **cmp** для файлов сжатых LZMA
lzdiff	Запускает **diff** для файлов сжатых LZMA
lzegrep	Запускает **egrep** для файлов сжатых LZMA
lzfgrep	Запускает **fgrep** для файлов сжатых LZMA
lzgrep	Запускает **grep** для файлов сжатых LZMA
lzless	Запускает **less** для файлов сжатых LZMA
lzma	Сжимает или распаковывает файлы в формате LZMA
lzmadec	Небольшой и быстрый декодер для файлов сжатых LZMA.
lzmainfo	Показывает информацию, хранящуюся в заголовке сжатого файла LZMA
lzmore	Запускает **more** для файлов сжатых LZMA
unlzma	Распаковывает файлы в формате LZMA
unxz	Распаковывает файлы в формате XZ
xz	Сжимает или распаковывает файлы в формате XZ.
xzcat	Распаковывает в стандартный поток вывода
xzcmp	Запускает **cmp** для сжатых XZ файлов
xzdec	Небольшой и быстрый декодер для файлов сжатых XZ
xzdiff	Запускает **diff** для сжатых XZ файлов
xzegrep	Запускает **egrep** для сжатых XZ файлов
xzfgrep	Запускает **fgrep** для сжатых XZ файлов
xzgrep	Запускает **grep** для сжатых XZ файлов
xzless	Запускает **less** для сжатых XZ файлов
xzmore	Запускает **more** для сжатых XZ файлов
`liblzma`	Библиотека, реализующая сжатие данных без потерь с блочной сортировкой с использованием алгоритма Lempel-Ziv-Markov

8.9. Lz4-1.10.0

Lz4 - это алгоритм сжатия без потерь, обеспечивающий скорость сжатия более 500 МБ/с на ядро. Он оснащен чрезвычайно быстрым декодером со скоростью несколько ГБ/с. Lz4 может работать с Zstandard, позволяя обоим алгоритмам быстрее сжимать данные.

Приблизительное время сборки: 0.1 SBU
Требуемое дисковое пространство: 4.2 MB

8.9.1. Установка пакета Lz4

Скомпилируйте пакет:

```
make BUILD_STATIC=no PREFIX=/usr
```

Чтобы протестировать пакет, выполните:

```
make -j1 check
```

Установите пакет:

```
make BUILD_STATIC=no PREFIX=/usr install
```

8.9.2. Установка Lz4 - 32-битная версия

Выполните очистку после предыдущей сборки:

```
make clean
```

Скомпилируйте пакет:

```
CC="gcc -m32" make BUILD_STATIC=no
```

Установите пакет:

```
make BUILD_STATIC=no PREFIX=/usr LIBDIR=/usr/lib32 DESTDIR=$(pwd)/m32 install &&
cp -a m32/usr/lib32/* /usr/lib32/
```

8.9.3. Содержимое пакета Lz4

Установленные программы: lz4, lz4c (ссылка на lz4), lz4cat (ссылка на lz4) и unlz4 (ссылка на lz4)
Установленные библиотеки: liblz4.so

Краткое описание

lz4 Сжимает или распаковывает файлы, используя алгоритм LZ4

lz4c Сжимает файлы, используя LZ4

lz4cat Выводит содержимое файла, сжатого с использованием LZ4

unlz4 Распаковывает файлы, используя LZ4

`liblz4` Библиотека, реализующая сжатие данных без потерь с использованием алгоритма LZ4

8.10. Zstd-1.5.6

Zstandard — это алгоритм сжатия в реальном времени, обеспечивающий высокую степень сжатия. Он предлагает очень широкий диапазон компромиссов между сжатием и скоростью при поддержке очень быстрого декодера.

Приблизительное время сборки: 0.4 SBU
Требуемое дисковое пространство: 84 MB

8.10.1. Установка пакета Zstd

Скомпилируйте пакет:

```
make prefix=/usr
```

Примечание

В выходных данных теста есть несколько мест, выводящих сообщение 'failed'. Они ожидаемы, и только 'FAIL' является фактическим сбоем теста. Сбоев при тестировании быть не должно.

Чтобы протестировать пакет, выполните:

```
make check
```

Установите пакет:

```
make prefix=/usr install
```

Удалите статическую библиотеку:

```
rm -v /usr/lib/libzstd.a
```

8.10.2. Установка Zstd - 32-битная версия

Выполните очистку после предыдущей сборки:

```
make clean
```

Скомпилируйте пакет:

```
CC="gcc -m32" make prefix=/usr
```

Установите пакет:

```
make prefix=/usr DESTDIR=$PWD/DESTDIR install
cp -Rv DESTDIR/usr/lib/* /usr/lib32/
sed -e "/^libdir/s/lib$/lib32/" -i /usr/lib32/pkgconfig/libzstd.pc
rm -rf DESTDIR
```

8.10.3. Содержимое пакета Zstd

Установленные программы: zstd, zstdcat (ссылка на zstd), zstdgrep, zstdless, zstdmt (ссылка на zstd) и unzstd (ссылка на zstd)
Установленные библиотеки: libzstd.so

Краткое описание

zstd Сжимает или распаковывает файлы в формате ZSTD

zstdgrep Запускает **grep** на сжатых ZSTD файлах

zstdless Запускает **less** на сжатых ZSTD файлах

`libzstd` Библиотека, реализующая сжатие данных без потерь, с использованием алгоритма ZSTD

8.11. File-5.45

Пакет File содержит утилиту для определения типа указанного файла или файлов

Приблизительное время сборки: менее 0.1 SBU
Требуемое дисковое пространство: 17 MB

8.11.1. Установка пакета File

Подготовьте File к компиляции:

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

Скомпилируйте пакет:

```
make
```

Чтобы протестировать пакет, выполните:

```
make check
```

Установите пакет:

```
make install
```

8.11.2. Установка File - 32-битная версия

Выполните очистку после предыдущей сборки:

```
make distclean
```

Подготовьте File к компиляции:

```
CC="gcc -m32" ./configure \
    --prefix=/usr         \
    --libdir=/usr/lib32   \
    --host=i686-pc-linux-gnu
```

Скомпилируйте пакет:

```
make
```

Установите пакет:

```
make DESTDIR=$PWD/DESTDIR install
cp -Rv DESTDIR/usr/lib32/* /usr/lib32
rm -rf DESTDIR
```

8.11.3. Содержимое пакета File

Установленные программы: file
Установленные библиотеки: libmagic.so

Краткое описание

file Пытается классифицировать каждый указанный файл; он делает это, выполняя серию тестов — тесты файловой системы, тесты магических чисел и языковые тесты.

`libmagic` Содержит функции распознавания магических чисел используемые программой **file**

8.12. Readline-8.2.13

Пакет Readline представляет собой набор библиотек, предлагающих возможности редактирования прямо в командной строке и просмотра истории команд.

Приблизительное время сборки: менее 0.1 SBU
Требуемое дисковое пространство: 16 MB

8.12.1. Установка пакета Readline

Переустановка пакета Readline приводит к перемещению старых библиотек в <libraryname>.old. Обычно это не вызывает проблем, но в некоторых случаях могут возникать ошибки линковки с **ldconfig**. Этого можно избежать, выполнив следующие две команды sed:

```
sed -i '/MV.*old/d' Makefile.in
sed -i '/{OLDSUFF}/c:' support/shlib-install
```

Запретите использование библиотеки поиска списка жестко запрограммированных путей (rpath) в общих библиотеках. Этому пакету не требуется rpath для установки в стандартное местоположение, кроме того, rpath иногда может вызывать нежелательные эффекты или даже проблемы с безопасностью:

```
sed -i 's/-Wl,-rpath,[^ ]*//' support/shobj-conf
```

Подготовьте Readline к компиляции:

```
./configure --prefix=/usr    \
            --disable-static \
            --with-curses    \
            --docdir=/usr/share/doc/readline-8.2.13
```

Значение нового параметра configure:

`--with-curses`
Этот параметр сообщает Readline, что он может найти функции библиотеки termcap в библиотеке curses, а не в отдельной библиотеке termcap. Это позволит сгенерировать корректный файл `readline.pc` .

Скомпилируйте пакет:

```
make SHLIB_LIBS="-lncursesw"
```

Значение параметра make:

`SHLIB_LIBS="-lncursesw"`
Этот параметр принудительно линкует Readline с библиотекой `libncursesw`.

С этим пакетом не поставляется набор тестов.

Установите пакет:

```
make SHLIB_LIBS="-lncursesw" install
```

По желанию установите документацию:

```
install -v -m644 doc/*.{ps,pdf,html,dvi} /usr/share/doc/readline-8.2.13
```

8.12.2. Установка Readline - 32-битная версия

Выполните очистку после предыдущей сборки:

```
make distclean
```

Подготовьте Readline к компиляции:

```
CC="gcc -m32" ./configure \
    --host=i686-pc-linux-gnu      \
    --prefix=/usr                 \
    --libdir=/usr/lib32           \
    --disable-static              \
    --with-curses
```

Скомпилируйте пакет:

```
make SHLIB_LIBS="-lncursesw"
```

Установите пакет:

```
make SHLIB_LIBS="-lncursesw" DESTDIR=$PWD/DESTDIR install
cp -Rv DESTDIR/usr/lib32/* /usr/lib32
rm -rf DESTDIR
```

8.12.3. Содержимое пакета Readline

Установленные библиотеки: libhistory.so и libreadline.so

Созданные каталоги: /usr/include/readline и /usr/share/doc/readline-8.2.13

Краткое описание

`libhistory` Обеспечивает согласованный пользовательский интерфейс для вызова строк из истории

`libreadline` Предоставляет набор команд для управления текстом, введенным в интерактивном сеансе программы.

8.13. M4-1.4.19

Пакет M4 содержит макропроцессор.

Приблизительное время сборки: 0.3 SBU
Требуемое дисковое пространство: 48 MB

8.13.1. Установка пакета M4

Подготовьте M4 к компиляции:

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

Скомпилируйте пакет:

```
make
```

Чтобы протестировать пакет, выполните:

```
make check
```

Установите пакет:

```
make install
```

8.13.2. Содержимое пакета M4

Установленные программы: m4

Краткое описание

m4 Копирует указанные файлы, одновременно расширяя содержащиеся в них макросы. Эти макросы являются либо встроенными, либо определяемыми пользователем и могут принимать любое количество аргументов. Помимо выполнения макросов, **m4** имеет встроенные функции для включения указанных файлов, выполнения команд Unix, выполнения целочисленной арифметики, манипулирования текстом, рекурсии и т.д. Программа **m4** может использоваться либо как интерфейс к компилятору, либо как самостоятельный макропроцессор

8.14. Bc-6.7.6

Пакет Bc содержит язык для обработки чисел произвольной точности.

Приблизительное время сборки: менее 0.1 SBU
Требуемое дисковое пространство: 7.8 MB

8.14.1. Установка пакета Bc

Подготовьте Bc к компиляции:

```
CC=gcc ./configure --prefix=/usr -G -O3 -r
```

Значение параметров настройки:

CC=gcc
Этот параметр определяет используемый компилятор

-G
Пропускает часть тестов, которые не будут работать, пока не будет установлена программа bc.

-O3
Указывает используемый уровень оптимизации.

-r
Включает использование Readline для улучшения функции редактирования строк в bc.

Скомпилируйте пакет:

```
make
```

Чтобы протестировать пакет, выполните:

```
make test
```

Установите пакет:

```
make install
```

8.14.2. Содержимое пакета Bc

Установленные программы: bc и dc

Краткое описание

bc Калькулятор командной строки

dc Калькулятор командной строки с обратной польской нотацией

8.15. Flex-2.6.4

Пакет Flex содержит инструмент для генерации программ, распознающих заданные шаблоны в тексте

Приблизительное время сборки:	0.1 SBU
Требуемое дисковое пространство:	33 MB

8.15.1. Установка пакета Flex

Подготовьте Flex к компиляции:

```
./configure --prefix=/usr \
            --docdir=/usr/share/doc/flex-2.6.4 \
            --disable-static
```

Скомпилируйте пакет:

```
make
```

Для тестирования пакета (около 0,5 SBU) выполните:

```
make check
```

Установите пакет:

```
make install
```

Некоторые программы пока не знают о **flex** и пытаются запустить его предшественника - **lex**. Чтобы обеспечить их работоспособность, создайте символическую ссылку **lex**, которая запускает `flex` в режиме эмуляции **lex**, а также создайте символическую ссылку на справочную страницу **lex**:

```
ln -sv flex   /usr/bin/lex
ln -sv flex.1 /usr/share/man/man1/lex.1
```

8.15.2. Содержимое пакета Flex

Установленные программы:	flex, flex++ (ссылка на flex), и lex (ссылка на flex)
Установленные библиотеки:	libfl.so
Созданные каталоги:	/usr/share/doc/flex-2.6.4

Краткое описание

flex	Инструмент для создания программ, распознающих текст по шаблону; это позволяет гибко указывать правила поиска паттернов, устраняя необходимость разработки специализированной программы.
flex++	Расширение flex используется для генерации кода и классов C++. Является символической ссылкой на **flex**
lex	Символическая ссылка, запускает **flex** в режиме эмуляции **lex**
`libfl`	Библиотека `flex`

8.16. Tcl-8.6.14

Пакет Tcl содержит Tool Command Language, надежный скриптовый язык общего назначения. Пакет Expect написан на языке Tcl (произносится как "тикл").

Приблизительное время сборки: 3.2 SBU
Требуемое дисковое пространство: 91 MB

8.16.1. Установка пакета Tcl

Этот пакет и следующие два (Expect и DejaGNU) устанавливаются для поддержки возможности тестирования Binutils, GCC и других пакетов. Установка трех пакетов для целей тестирования может показаться избыточной, но вы будете чувствовать себя увереннее, когда знаете, что наиболее важные инструменты работают правильно.

Подготовьте Tcl к компиляции:

```
SRCDIR=$(pwd)
cd unix
./configure --prefix=/usr           \
            --mandir=/usr/share/man \
            --disable-rpath
```

Значение новых параметров конфигурации:

`--disable-rpath`
Этот параметр предотвращает использование библиотеки поиска списка жестко запрограммированных путей (rpath) в двоичных исполняемых файлах и общих библиотеках. Этому пакету не требуется rpath для установки в стандартное местоположение, кроме того, rpath иногда может вызывать нежелательные эффекты или даже проблемы с безопасностью.

Соберите пакет:

```
make

sed -e "s|$SRCDIR/unix|/usr/lib|" \
    -e "s|$SRCDIR|/usr/include|"  \
    -i tclConfig.sh

sed -e "s|$SRCDIR/unix/pkgs/tdbc1.1.7|/usr/lib/tdbc1.1.7|" \
    -e "s|$SRCDIR/pkgs/tdbc1.1.7/generic|/usr/include|"    \
    -e "s|$SRCDIR/pkgs/tdbc1.1.7/library|/usr/lib/tcl8.6|" \
    -e "s|$SRCDIR/pkgs/tdbc1.1.7|/usr/include|"            \
    -i pkgs/tdbc1.1.7/tdbcConfig.sh

sed -e "s|$SRCDIR/unix/pkgs/itcl4.2.4|/usr/lib/itcl4.2.4|" \
    -e "s|$SRCDIR/pkgs/itcl4.2.4/generic|/usr/include|"    \
    -e "s|$SRCDIR/pkgs/itcl4.2.4|/usr/include|"            \
    -i pkgs/itcl4.2.4/itclConfig.sh

unset SRCDIR
```

Различные инструкции «sed» после команды «make» удаляют ссылки на каталог сборки из файлов конфигурации и заменяют их на созданные каталоги. Это необязательно для остальной части LFS, но может понадобиться в случае, когда пакет, собранный позже, использует Tcl.

Чтобы протестировать пакет, выполните:

```
make test
```

Установите пакет:

```
make install
```

Сделайте установленную библиотеку доступной для записи, чтобы позже можно было удалить отладочные символы:

```
chmod -v u+w /usr/lib/libtcl8.6.so
```

Установите заголовочные файлы Tcl. Они потребуются для следующего пакета - Expect.

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

Теперь создайте необходимую символическую ссылку:

```
ln -sfv tclsh8.6 /usr/bin/tclsh
```

Переименуйте справочную страницу, которая конфликтует со справочной страницей Perl:

```
mv /usr/share/man/man3/{Thread,Tcl_Thread}.3
```

При необходимости установите документацию, выполнив следующие команды:

```
cd ..
tar -xf ../tcl8.6.14-html.tar.gz --strip-components=1
mkdir -v -p /usr/share/doc/tcl-8.6.14
cp -v -r  ./html/* /usr/share/doc/tcl-8.6.14
```

8.16.2. Содержимое пакета Tcl

Установленные программы:	tclsh (ссылка на tclsh8.6) и tclsh8.6
Установленные библиотеки:	libtcl8.6.so и libtclstub8.6.a

Краткое описание

tclsh8.6	Командная оболочка Tcl
tclsh	Ссылка на tclsh8.6
`libtcl8.6.so`	Библиотека Tcl
`libtclstub8.6.a`	Библиотека-заглушка Tcl

8.17. Expect-5.45.4

Пакет Expect содержит инструменты для автоматизации работы интерактивных приложений, таких как **telnet**, **ftp**, **passwd**, **fsck**, **rlogin** и **tip**, с помощью скриптовых диалогов и макросов. Кроме того Expect полезен для тестирования перечисленных выше приложений, а также для решения сложных задач взаимодействия с другими средствами. Фреймворк DejaGnu написан на языке Expect.

Приблизительное время сборки: 0.2 SBU
Требуемое дисковое пространство: 3.9 MB

8.17.1. Установка пакета Expect

Expect для работы необходим псевдотерминал(PTY). Убедитесь, что PTY работают правильно в среде chroot, выполнив простой тест:

```
python3 -c 'from pty import spawn; spawn(["echo", "ok"])'
```

Эта команда должна вывести `ok`. Если вместо этого вывод содержит `OSError: out of pty devices`, то среда не настроена для правильной работы PTY. Вам необходимо выйти из среды chroot, ещё раз прочитать Раздел 7.3, «Подготовка виртуальных файловых систем ядра» и убедиться, что файловая система `devpts` (и другие файловые системы виртуального ядра) смонтирована правильно. Затем повторно войдите в среду chroot, следуя инструкции Раздел 7.4, «Вход в окружение Chroot». Эту проблему необходимо решить, прежде чем вы продолжите, иначе наборы тестов, зависимые от Expect (например, наборы тестов Bash, Binutils, GCC, GDBM и, конечно, самого Expect), потерпят фатальный сбой, а также могут произойти другие незначительные сбои.

Теперь внесите некоторые изменения, чтобы разрешить использование пакета с gcc-14.1 или более поздней версией:

```
patch -Np1 -i ../expect-5.45.4-gcc14-1.patch
```

Подготовьте Expect к компиляции:

```
./configure --prefix=/usr           \
            --with-tcl=/usr/lib     \
            --enable-shared         \
            --disable-rpath         \
            --mandir=/usr/share/man \
            --with-tclinclude=/usr/include
```

Значение параметров настройки:

`--with-tcl=/usr/lib`
Этот параметр необходим для указания **configure** где находится скрипт **tclConfig.sh**.

`--with-tclinclude=/usr/include`
Этот параметр явно указывает Expect, где искать внутренние заголовки Tcl.

Соберите пакет:

```
make
```

Чтобы протестировать пакет, выполните:

```
make test
```

Установите пакет:

```
make install
ln -svf expect5.45.4/libexpect5.45.4.so /usr/lib
```

8.17.2. Содержимое пакета Expect

Установленные программы:	expect
Установленные библиотеки:	libexpect5.45.4.so

Краткое описание

expect	Взаимодействует с другими интерактивными программами в соответствии со сценарием
`libexpect-5.45.4.so`	Содержит функции, которые позволяют использовать Expect в качестве расширения Tcl или непосредственно из C или C++ (без Tcl).

8.18. DejaGNU-1.6.3

Пакет DejaGnu содержит фреймворк для запуска наборов тестов на инструментах GNU. Он написан на **expect**, который в свою очередь использует Tcl (командный язык инструментов).

Приблизительное время сборки: менее 0.1 SBU
Требуемое дисковое пространство: 6.9 MB

8.18.1. Установка пакета DejaGNU

Разработчики рекомендуют собирать DejaGNU в отдельном каталоге:

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

Подготовьте DejaGNU к компиляции:

```
../configure --prefix=/usr
makeinfo --html --no-split -o doc/dejagnu.html ../doc/dejagnu.texi
makeinfo --plaintext       -o doc/dejagnu.txt  ../doc/dejagnu.texi
```

Чтобы протестировать пакет, выполните:

```
make check
```

Установите пакет:

```
make install
install -v -dm755  /usr/share/doc/dejagnu-1.6.3
install -v -m644   doc/dejagnu.{html,txt} /usr/share/doc/dejagnu-1.6.3
```

8.18.2. Содержимое пакета DejaGNU

Установленные программы: dejagnu и runtest

Short Descriptions

dejagnu Вспомогательная программа запуска команд DejaGNU

runtest Скрипт-обертка, который находит соответствующую оболочку **expect** и запускает DejaGnu

8.19. Pkgconf-2.3.0

Пакет pkgconf является преемником pkg-config и содержит инструмент, который позволяет передавать пути установки или пути к библиотекам для инструментов сборки на этапе настройки (configure) и сборки(make) пакетов.

Приблизительное время сборки: менее 0.1 SBU
Требуемое дисковое пространство: 4.7 MB

8.19.1. Установка пакета Pkgconf

Подготовьте Pkgconf к компиляции:

```
./configure --prefix=/usr              \
            --disable-static           \
            --docdir=/usr/share/doc/pkgconf-2.3.0
```

Скомпилируйте пакет:

```
make
```

Установите пакет:

```
make install
```

Для обеспечения совместимости с исходным Pkg-config, создайте две символические ссылки:

```
ln -sv pkgconf   /usr/bin/pkg-config
ln -sv pkgconf.1 /usr/share/man/man1/pkg-config.1
```

8.19.2. Содержимое пакета Pkgconf

Установленные программы: pkgconf, pkg-config (ссылка на pkgconf) и bomtool
Установленные библиотеки: libpkgconf.so
Созданные каталоги: /usr/share/doc/pkgconf-2.3.0

Краткое описание

pkgconf Возвращает метаданные указанной библиотеки или пакета

bomtool Генерирует спецификацию программного обеспечения из файлов pkg-config с расширением .pc

`libpkgconf` Содержит большую часть функций pkgconf, позволяя другим инструментам, таким как IDE и компиляторы, использовать его фреймворки

8.20. Binutils-2.43.1

Пакет Binutils содержит компоновщик, ассемблер и другие инструменты для работы с объектными файлами.

Приблизительное время сборки:	2.0 SBU
Требуемое дисковое пространство:	2.7 GB

8.20.1. Установка пакета Binutils

Документация Binutils рекомендует выполнять компиляцию в отдельном каталоге:

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

Подготовьте Binutils к компиляции:

```
../configure --prefix=/usr       \
             --sysconfdir=/etc   \
             --enable-gold       \
             --enable-ld=default \
             --enable-plugins    \
             --enable-shared     \
             --disable-werror    \
             --enable-64-bit-bfd \
             --with-system-zlib  \
             --enable-default-hash-style=gnu \
             --enable-multilib
```

Значение параметров настройки:

`--enable-gold`
Собирает компоновщик gold и устанавливает его как ld.gold (вместе с компоновщиком по умолчанию).

`--enable-ld=default`
Собирает оригинальный компоновщик bfd и устанавливает его как ld (компоновщик по умолчанию) и как ld.bfd

`--enable-plugins`
Включает поддержку плагинов для компоновщика.

`--with-system-zlib`
Использовать установленную библиотеку zlib вместо сборки собственной.

`--enable-multilib`
Включает поддержку multilib в bintutils.

Скомпилируйте пакет:

```
make tooldir=/usr
```

Значение параметра make:

`tooldir=/usr`
Обычно для tooldir (каталога, в котором будут расположены исполняемые файлы) установлено значение `$(exec_prefix)/$(target_alias)`. Например, машины x86_64 преобразуют это значение в `/usr/x86_64-unknown-linux-gnu`. Поскольку это пользовательская система, то целевой каталог в `/usr` не требуется. Параметр `$(exec_prefix)/$(target_alias)` использовался, если бы система применялась для кросс-компиляции (например, при компиляции пакета на компьютере Intel, который генерирует код, который может быть выполнен на компьютерах PowerPC).

Важно

Набор тестов для Binutils в этом разделе считается критически важным. Ни в коем случае не пропускайте его.

Выполните тестирование:

```
make -k check
```

Чтобы получить список неудачных тестов, запустите:

```
grep '^FAIL:' $(find -name '*.log')
```

Двенадцать тестов завершаются неудачно в наборе тестов gold, когда GCC собирается с параметрами `--enable-default-pie` и `--enable-default-ssp`.

Также известно, что завершаются неудачно три теста в пакете gprofng.

Установите пакет:

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

Удалите бесполезные статические библиотеки:

```
rm -fv /usr/lib/lib{bfd,ctf,ctf-nobfd,gprofng,opcodes,sframe}.a
```

8.20.2. Содержимое пакета Binutils

Установленные программы:	addr2line, ar, as, c++filt, dwp, elfedit, gprof, gprofng, ld, ld.bfd, ld.gold, nm, objcopy, objdump, ranlib, readelf, size, strings и strip
Установленные библиотеки:	libbfd.so, libctf.so, libctf-nobfd.so, libgprofng.so, libopcodes.so и libsframe.so
Созданные каталоги:	/usr/lib/ldscripts

Краткое описание

addr2line	Переводит адреса программ в имена файлов и номера строк; учитывая адрес и имя исполняемого файла, использует отладочную информацию в исполняемом файле, для определения файла исходного кода и номера строки, ассоциированной с адресом
ar	Создаёт, изменяет и распаковывает архивы
as	Ассемблер, который собирает результат работы **gcc** в объектные файлы
c++filt	Используется компоновщиком для исправления символов C++ и Java и предотвращения конфликтов перегруженных функций.
dwp	Утилита для упаковки DWARF
elfedit	Обновляет ELF заголовки в ELF файлах
gprof	Отображает в графическом виде информацию о профилировании
gprofng	Собирает и анализирует данные о производительности
ld	Компоновщик, который объединяет несколько объектных и архивных файлов в один файл, перемещая их данные и связывая символическими ссылками
ld.gold	Урезанная версия ld, которая поддерживает только формат объектных файлов elf
ld.bfd	Жесткая ссылка на **ld**
nm	Выводит список символов, используемых в данном объектном файле
objcopy	Преобразует один тип объектного файла в другой

objdump	Отображает информацию о данном объектном файле; можно указать параметры, определяющие, какая конкретно информация будет отображаться. Отображаемая информация полезна для программистов, которые работают над инструментами, используемыми при компиляции
ranlib	Создает индекс содержимого архива и сохраняет его в архиве; в индексе перечислены все символы, определенные в перемещаемых объектных файлах, содержащихся в архиве
readelf	Отображает информацию о двоичных файлах типа ELF
size	Отображает размеры секций и общий размер указанных объектных файлов
strings	Выводит для каждого указанного файла последовательности печатаемых символов, которые имеют по крайней мере указанную длину (по умолчанию четыре); для объектных файлов по умолчанию печатаются только строки из секций инициализации и загрузки, в то время как для других файлов он сканирует весь файл.
strip	Удаляет символы из объектных файлов
`libbfd`	Библиотека дескрипторов двоичных файлов
`libctf`	Библиотека отладки формата Compat ANSI-C Type
`libctf-nobfd`	Вариант libctf, не использующий функциональность libbfd.
`libgprofng`	Библиотека, содержащая большинство подпрограмм, используемых **gprofng**
`libopcodes`	Библиотека для работы с опкодами — «читаемыми» версиями инструкций для процессора. Используется для сборки таких утилит как **objdump**
`libsframe`	Библиотека для поддержки обратной онлайн-трассировки с использованием простого декодера разделов .sframe.

8.21. GMP-6.3.0

Пакет GMP содержит математические библиотеки. Они содержат полезные функции для арифметики с произвольной точностью.

Приблизительное время сборки: 0.3 SBU

Требуемое дисковое пространство: 54 MB

8.21.1. Установка пакета GMP

Примечание

Если вы выполняете сборку для 32-разрядной архитектуры x86, но ваш процессор, способен выполнять 64-разрядный код, *и* вы указали в переменных окружения `CFLAGS`, скрипт configure попытается выполнить настройку для 64-разрядной системы и завершится ошибкой. Чтобы избежать этого, необходимо вызвать команду configure с приведенным ниже параметром

```
ABI=32 ./configure ...
```

Примечание

Настройки GMP по умолчанию собирают библиотеки, оптимизированные для процессора хоста. Если требуются библиотеки, подходящие для процессоров с меньшей производительностью, чем у процессора хоста, можно собрать общие библиотеки, добавив параметр `--host=none-linux-gnu` в команде **configure**.

Подготовьте GMP к компиляции:

```
./configure --prefix=/usr    \
            --enable-cxx     \
            --disable-static \
            --docdir=/usr/share/doc/gmp-6.3.0
```

Значение новых параметров настройки:

`--enable-cxx`
Этот параметр включает поддержку C++

`--docdir=/usr/share/doc/gmp-6.3.0`
Эта переменная указывает местоположение для документации.

Скомпилируйте пакет и сгенерируйте HTML-документацию:

```
make
make html
```

Важно

Набор тестов для GMP в этом разделе считается критически важным. Ни в коем случае не пропускайте его.

Проверьте результаты:

```
make check 2>&1 | tee gmp-check-log
```

Внимание

Код в GMP сильно оптимизирован для процессора, на котором он собран. Иногда код, определяющий процессор, неверно определяет возможности системы, и в тестах или других приложениях, использующих библиотеки gmp, возникают ошибки с сообщением `Illegal instruction`. В этом случае gmp следует переконфигурировать с параметром `--host=none-linux-gnu` и пересобрать.

Убедитесь, что все 199 тестов в наборе пройдены. Проверьте результат, выполнив следующую команду:

```
awk '/# PASS:/{total+=$3} ; END{print total}' gmp-check-log
```

Установите пакет и его документацию:

```
make install
make install-html
```

8.21.2. Установка GMP - 32-битная версия

Выполните очистку после предыдущей сборки:

```
make distclean
```

Универсальные библиотеки можно создать, выполнив следующие действия:

```
cp -v configfsf.guess config.guess
cp -v configfsf.sub   config.sub
```

Подготовьте GMP к компиляции:

```
ABI="32" \
CFLAGS="-m32 -O2 -pedantic -fomit-frame-pointer -mtune=generic -march=i686" \
CXXFLAGS="$CFLAGS" \
PKG_CONFIG_PATH="/usr/lib32/pkgconfig" \
./configure                       \
    --host=i686-pc-linux-gnu      \
    --prefix=/usr                 \
    --disable-static              \
    --enable-cxx                  \
    --libdir=/usr/lib32           \
    --includedir=/usr/include/m32/gmp
```

Значение новых параметров настройки:

`--includedir=/usr/include/m32/gmp`
Некоторые определения в gmp.h различаются в зависимости от архитектуры, но имеют одно и то же имя. Поэтому заголовки должны быть отделены друг от друга.

Скомпилируйте пакет:

```
sed -i 's/$(exec_prefix)\/include/$\(includedir\)/' Makefile
make
```

Важно

Набор тестов для GMP, приведенный в этом разделе, считается критически важным. Не пропускайте его ни при каких обстоятельствах.

Проверьте результаты:

```
make check 2>&1 | tee gmp-check-log
```

Убедитесь, что все 199 тестов в наборе пройдены. Проверьте результат, выполнив следующую команду:

```
awk '/# PASS:/{total+=$3} ; END{print total}' gmp-check-log
```

Установите пакет:

```
make DESTDIR=$PWD/DESTDIR install
cp -Rv DESTDIR/usr/lib32/* /usr/lib32
cp -Rv DESTDIR/usr/include/m32/* /usr/include/m32/
rm -rf DESTDIR
```

8.21.3. Содержимое пакета GMP

Установленные библиотеки: libgmp.so и libgmpxx.so
Созданные каталоги: /usr/share/doc/gmp-6.3.0

Краткое описание

`libgmp` Содержит точные математические функции

`libgmpxx` Содержит точные математические функции C++

8.22. MPFR-4.2.1

Пакет MPFR содержит функции для двоичных вычислений с плавающей запятой произвольной точности.

Приблизительное время сборки: 0.2 SBU
Требуемое дисковое пространство: 43 MB

8.22.1. Установка пакета MPFR

Подготовьте MPFR к компиляции:

```
./configure --prefix=/usr        \
            --disable-static     \
            --enable-thread-safe \
            --docdir=/usr/share/doc/mpfr-4.2.1
```

Скомпилируйте пакет и сгенерируйте HTML-документацию:

```
make
make html
```

Важно

Набор тестов для MPFR в этом разделе считается критически важным. Ни в коем случае не пропускайте его.

Выполните тестирование и убедитесь, что все 198 тестов пройдены:

```
make check
```

Установите пакет и документацию к нему:

```
make install
make install-html
```

8.22.2. Содержимое пакета MPFR

Установленные библиотеки: libmpfr.so
Созданные каталоги: /usr/share/doc/mpfr-4.2.1

Краткое описание

`libmpfr` Содержит математические функции с произвольной точностью

8.23. MPC-1.3.1

Пакет MPC содержит библиотеку для арифметики комплексных чисел с высокой точностью и правильным округлением результата.

Приблизительное время сборки: 0.1 SBU
Требуемое дисковое пространство: 22 MB

8.23.1. Установка пакета MPC

Подготовьте MPC к компиляции:

```
./configure --prefix=/usr    \
            --disable-static \
            --docdir=/usr/share/doc/mpc-1.3.1
```

Скомпилируйте пакет и сгенерируйте HTML-документацию:

```
make
make html
```

Чтобы протестировать пакет, выполните:

```
make check
```

Установите пакет и документацию к нему:

```
make install
make install-html
```

8.23.2. Содержимое пакета MPC

Установленные библиотеки: libmpc.so
Созданные каталоги: /usr/share/doc/mpc-1.3.1

Краткое описание

`libmpc` Содержит сложные математические функции

8.24. ISL-0.26

ISL - это потокобезопасная библиотека C для манипулирования множествами и отношениями целых точек, связанных линейными ограничениями.

Приблизительное время сборки: 0.1 SBU
Требуемое дисковое пространство: 20 MB

8.24.1. Установка пакета ISL

Подготовьте ISL к компиляции:

```
./configure --prefix=/usr    \
            --disable-static \
            --docdir=/usr/share/doc/isl-0.26
```

Значение новых параметров настройки:

`--docdir=/usr/share/doc/isl-0.26`
Эта переменная указывает правильное местоположение документации.

Скомпилируйте пакет:

```
make
```

Установите пакет и документацию к нему:

```
make install
install -vd /usr/share/doc/isl-0.26
install -m644 doc/{CodingStyle,manual.pdf,SubmittingPatches,user.pod} \
        /usr/share/doc/isl-0.26
```

Наконец, переместите файл в правильное местоположение:

```
mkdir -pv /usr/share/gdb/auto-load/usr/lib
mv -v /usr/lib/libisl*gdb.py /usr/share/gdb/auto-load/usr/lib
```

8.24.2. Содержимое пакета ISL

Установленные библиотеки: libisl.so
Созданные каталоги: /usr/share/doc/isl-0.26

Краткое описание

`libisl` Содержит функции манипулирования целочисленными наборами

8.25. Attr-2.5.2

Пакет Attr содержит утилиты для управления расширенными атрибутами объектов файловой системы.

Приблизительное время сборки: менее 0.1 SBU
Требуемое дисковое пространство: 4.1 MB

8.25.1. Установка пакета Attr

Подготовьте Attr к компиляции:

```
./configure --prefix=/usr     \
            --disable-static  \
            --sysconfdir=/etc \
            --docdir=/usr/share/doc/attr-2.5.2
```

Скомпилируйте пакет:

```
make
```

Тесты необходимо запускать в файловой системе, которая поддерживает расширенные атрибуты, например, ext2, ext3 или ext4. Чтобы протестировать пакет, выполните:

```
make check
```

Установите пакет:

```
make install
```

8.25.2. Установка Attr - 32-битная версия

Выполните очистку после предыдущей сборки:

```
make distclean
```

Подготовьте Attr к компиляции:

```
CC="gcc -m32" ./configure \
    --prefix=/usr         \
    --disable-static      \
    --sysconfdir=/etc     \
    --libdir=/usr/lib32   \
    --host=i686-pc-linux-gnu
```

Скомпилируйте пакет:

```
make
```

Установите пакет:

```
make DESTDIR=$PWD/DESTDIR install
cp -Rv DESTDIR/usr/lib32/* /usr/lib32
rm -rf DESTDIR
```

8.25.3. Содержимое пакета Attr

Установленные программы: attr, getfattr, и setfattr
Установленные библиотеки: libattr.so
Созданные каталоги: /usr/include/attr и /usr/share/doc/attr-2.5.2

Краткое описание

attr	Расширяет атрибуты объектов файловой системы
getfattr	Получает расширенные атрибуты объектов файловой системы
setfattr	Устанавливает расширенные атрибуты объектов файловой системы
`libattr`	Содержит библиотечные функции для управления расширенными атрибутами.

8.26. Acl-2.3.2

Пакет Acl содержит утилиты для администрирования списков контроля доступа, которые используются для определения расширенных дискреционных прав доступа к файлам и каталогам.

Приблизительное время сборки: менее 0.1 SBU
Требуемое дисковое пространство: 6.3 MB

8.26.1. Установка пакета Acl

Подготовьте Acl к компиляции:

```
./configure --prefix=/usr         \
            --disable-static      \
            --docdir=/usr/share/doc/acl-2.3.2
```

Скомпилируйте пакет:

```
make
```

Тесты Acl необходимо запускать в файловой системе, поддерживающей списки контроля доступа, после сборки пакета Coreutils с использованием библиотек Acl. По желанию вернитесь к этому пакету и запустите **make check** после того, как будет собран пакет Coreutils.

Установите пакет:

```
make install
```

8.26.2. Установка Acl - 32-битная версия

Выполните очистку после предыдущей сборки:

```
make distclean
```

Подготовьте Acl к компиляции:

```
CC="gcc -m32" ./configure \
    --prefix=/usr         \
    --disable-static      \
    --libdir=/usr/lib32   \
    --libexecdir=/usr/lib32   \
    --host=i686-pc-linux-gnu
```

Скомпилируйте пакет:

```
make
```

Установите пакет:

```
make DESTDIR=$PWD/DESTDIR install
cp -Rv DESTDIR/usr/lib32/* /usr/lib32
rm -rf DESTDIR
```

8.26.3. Содержимое пакета Acl

Установленные программы: chacl, getfacl, и setfacl
Установленные библиотеки: libacl.so
Созданные каталоги: /usr/include/acl и /usr/share/doc/acl-2.3.2

Краткое описание

chacl	Изменяет список контроля доступа файла или каталога
getfacl	Получает списки контроля доступа файла
setfacl	Устанавливает списки контроля доступа к файлам
`libacl`	Содержит библиотечные функции для управления списками контроля доступа.

8.27. Libcap-2.70

Пакет Libcap реализует интерфейсы пользовательского пространства для возможностей POSIX 1003.1e, доступных в ядрах Linux. Эти возможности разделяют полномочия суперпользователя root на набор отдельных привилегий.

Приблизительное время сборки: менее 0.1 SBU
Требуемое дисковое пространство: 2.9 MB

8.27.1. Установка пакета Libcap

Запретите установку статических библиотек:

```
sed -i '/install -m.*STA/d' libcap/Makefile
```

Скомпилируйте пакет:

```
make prefix=/usr lib=lib
```

Значение опции make:

lib=lib
Этот параметр устанавливает библиотеки в каталог `/usr/lib` , а не `/usr/lib64` на x86_64. На x86 это никак не влияет.

Чтобы протестировать пакет, выполните:

```
make test
```

Установите пакет:

```
make prefix=/usr lib=lib install
```

8.27.2. Установка Libcap - 32-битная версия

Выполните очистку после предыдущей сборки:

```
make distclean
```

Скомпилируйте пакет:

```
make CC="gcc -m32 -march=i686"
```

Установите пакет:

```
make CC="gcc -m32 -march=i686" lib=lib32 prefix=$PWD/DESTDIR/usr -C libcap install
cp -Rv DESTDIR/usr/lib32/* /usr/lib32
sed -e "s|^libdir=.*|libdir=/usr/lib32|" -i /usr/lib32/pkgconfig/lib{cap,psx}.pc
chmod -v 755 /usr/lib32/libcap.so.2.70
rm -rf DESTDIR
```

8.27.3. Содержимое пакета Libcap

Установленные программы: capsh, getcap, getpcaps и setcap
Установленные библиотеки: libcap.so и libpsx.so

Краткое описание

capsh Обёртка к оболочке для исследования и ограничения поддержки возможностей Linux

getcap	Проверяет возможности файлов
getpcaps	Отображает возможности запрашиваемого процесса (процессов)
setcap	Устанавливает возможности файлов
`libcap`	Содержит функции для управления возможностями POSIX 1003.1e.
`libpsx`	Содержит функции для поддержки семантики POSIX для системных вызовов, связанных с библиотекой pthread

8.28. Libxcrypt-4.4.36

Пакет Libxcrypt содержит современную библиотеку для одностороннего хэширования паролей.

Приблизительное время сборки: 0.1 SBU
Требуемое дисковое пространство: 12 MB

8.28.1. Установка пакета Libxcrypt

Подготовьте Libxcrypt к компиляции:

```
./configure --prefix=/usr                \
            --enable-hashes=strong,glibc \
            --enable-obsolete-api=no     \
            --disable-static             \
            --disable-failure-tokens
```

Значение новых параметров настройки:

`--enable-hashes=strong,glibc`
Создает хэши, используя надежные алгоритмы хэширования, рекомендуемые для безопасности, и алгоритмы хэширования, предоставляемые традиционной библиотекой Glibc `libcrypt` для совместимости.

`--enable-obsolete-api=no`
Отключает устаревшие функции API. Они не нужны для современной системы Linux, собранной из исходного кода.

`--disable-failure-tokens`
Отключает признак токена сбоя. Он необходим для совместимости с традиционными хэш-библиотеками некоторых платформ, но система Linux, основанная на Glibc, в нем не нуждается.

Скомпилируйте пакет:

```
make
```

Чтобы протестировать пакет, выполните:

```
make check
```

Установите пакет:

```
make install
```

Примечание

С помощью приведенных выше инструкций были отключены устаревшие функции API, поскольку ни один пакет, установленный путем компиляции из исходного кода, не будет связываться с ними во время выполнения. Однако известные двоичные приложения, которые используют эти функции, требуют ABI версии 1. Если вам необходим этот функционал для какого-либо приложения, предоставляемого только в бинарном виде, или для совместимости с LSB, соберите пакет заново с помощью следующих команд:

```
make distclean
./configure --prefix=/usr                \
            --enable-hashes=strong,glibc \
            --enable-obsolete-api=glibc  \
            --disable-static             \
            --disable-failure-tokens
make
cp -av --remove-destination .libs/libcrypt.so.1* /usr/lib
```

8.28.2. Установка Libxcrypt - 32-битная версия

Выполните очистку после предыдущей сборки:

```
make distclean
```

Подготовьте Libxcrypt к компиляции:

```
CC="gcc -m32" \
./configure --prefix=/usr                \
            --host=i686-pc-linux-gnu     \
            --libdir=/usr/lib32          \
            --enable-hashes=strong,glibc \
            --enable-obsolete-api=glibc  \
            --disable-static             \
            --disable-failure-tokens
```

Скомпилируйте пакет:

```
make
```

Установите пакет:

```
cp -av .libs/libcrypt.so* /usr/lib32/ &&
make install-pkgconfigDATA &&
ln -svf libxcrypt.pc /usr/lib32/pkgconfig/libcrypt.pc
```

8.28.3. Содержимое пакета Libxcrypt

Установленные библиотеки: libcrypt.so

Краткое описание

`libcrypt` Содержит функции для хэширования паролей

8.29. Shadow-4.16.0

Пакет Shadow содержит программы для безопасной обработки паролей.

Приблизительное время сборки: 0.1 SBU

Требуемое дисковое пространство: 112 MB

8.29.1. Установка пакета Shadow

Важно

Если вы установили Linux-PAM, вам следует перейти *на страницу BLFS shadow* вместо этой страницы, чтобы собрать (пересобрать, или обновить) shadow.

Примечание

Если вы хотите принудительно использовать надежные пароли, обратитесь к инструкции *https://mirror.linuxfromscratch.ru/blfs/view/12.2/postlfs/cracklib.html* для установки CrackLib перед сборкой. Затем добавьте параметр `--with-libcrack` в приведенную ниже команду **configure**.

Отключите установку **groups** и ее справочных страниц, так как Coreutils предоставляет версию лучше. Кроме того, запретите установку страниц руководств, так как они были установлены в Раздел 8.3, «Man-pages-6.9.1»:

```
sed -i 's/groups$(EXEEXT) //' src/Makefile.in
find man -name Makefile.in -exec sed -i 's/groups\.1 / /'   {} \;
find man -name Makefile.in -exec sed -i 's/getspnam\.3 / /' {} \;
find man -name Makefile.in -exec sed -i 's/passwd\.5 / /'   {} \;
```

Вместо используемого по умолчанию метода *crypt*, используйте более безопасный метод шифрования паролей *YESCRYPT*, который также позволяет использовать пароли длиннее 8 символов. Также необходимо изменить устаревшее местоположение для почтовых ящиков пользователей `/var/spool/mail`, которое Shadow использует по умолчанию, на используемое в настоящее время `/var/mail`. И удалите `/bin` и `/sbin` из `PATH`, поскольку они являются просто символическими ссылками на их аналоги в `/usr`.

Примечание

Если вы по какой-либо причине хотите включить `/bin` и/или `/sbin` в `PATH`, измените `PATH` в файле `.bashrc` после сборки LFS.

```
sed -e 's:#ENCRYPT_METHOD DES:ENCRYPT_METHOD YESCRYPT:' \
    -e 's:/var/spool/mail:/var/mail:'                   \
    -e '/PATH=/{s@/sbin:@@;s@/bin:@@}'                  \
    -i etc/login.defs
```

Примечание

Если вы решили собрать Shadow с поддержкой Cracklib, выполните эту команду:

```
sed -i 's:DICTPATH.*:DICTPATH\t/lib/cracklib/pw_dict:' etc/login.defs
```

Подготовьте Shadow к компиляции:

```
touch /usr/bin/passwd
./configure --sysconfdir=/etc \
            --disable-static  \
            --with-{b,yes}crypt \
            --without-libbsd    \
            --with-group-name-max-length=32
```

Значение новых параметров конфигурации:

touch /usr/bin/passwd

Файл `/usr/bin/passwd` должен существовать, потому что его местоположение жестко задано в некоторых программах; если он не существует, скрипт установки создаст его не в том месте.

`--with-{b,yes}crypt`

Оболочка расширяет это значение до двух параметров: *`--with-bcrypt`* и *`--with-yescrypt`*. Они позволяют Shadow использовать алгоритмы Bcrypt и Yescrypt, реализованные в Libxcrypt, для хеширования паролей. Эти алгоритмы более безопасны (в частности, гораздо более устойчивы к атакам с использованием графических процессоров), чем традиционные алгоритмы SHA.

`--with-group-name-max-length=32`

Максимально допустимая длина имени пользователя составляет 32 символа. Сделайте такую же длину для названия группы.

`--without-libbsd`

Не используйте функцию readpassphrase из иблиотеки libbsd, которой нет в LFS. Вместо этого используйте внутреннюю копию.

Скомпилируйте пакет:

```
make
```

С этим пакетом не поставляется набор тестов.

Установите пакет:

```
make exec_prefix=/usr install
make -C man install-man
```

8.29.2. Настройка Shadow

Этот пакет содержит утилиты для добавления, изменения и удаления пользователей и групп; установки и изменения их паролей; и выполнения других задач администрирования. Полное объяснение того, что означает *password shadowing*, см. в файле doc/HOWTO в дереве распакованных исходных текстов. При использовании Shadow имейте в виду, что программы, которым необходимо проверять пароли (дисплейные менеджеры, FTP-программы, демоны pop3 и т.д.), должны быть совместимы с Shadow. То есть они должны уметь работать с теневыми паролями.

Чтобы включить поддержку теневых паролей, выполните следующую команду:

```
pwconv
```

Чтобы включить использование теневых паролей для групп, запустите:

```
grpconv
```

Конфигурация Shadow по умолчанию для утилиты **useradd** имеет несколько особенностей, требующих пояснения. Во-первых, по умолчанию утилита **useradd** создает пользователя и группу с тем же названием, что и имя пользователя. По умолчанию, идентификатора пользователя (UID) и идентификатора группы (GID) начинаются с 1000. Это означает, что если вы не передадите дополнительные параметры в **useradd**, каждый

пользователь будет членом уникальной группы в системе. Если такое поведение нежелательно, вам нужно передать один из параметров *-g* или *-N* в **useradd** или изменить настройку *USERGROUPS_ENAB* в файле */etc/login.defs*. Смотрите справочную страницу *useradd(8)* для получения дополнительной информации.

Во-вторых, чтобы изменить параметры по умолчанию, необходимо создать файл */etc/default/useradd* и настроить его в соответствии с вашими потребностями. Создайте его:

```
mkdir -p /etc/default
useradd -D --gid 999
```

Пояснения к параметрам /etc/default/useradd

GROUP=999

Этот параметр задает начальный номер группы, используемых в файле */etc/group*. Значение 999 берется из приведенного выше параметра *--gid*. Вы можете установить любое значение. Обратите внимание, что **useradd** никогда не будет повторно использовать UID или GID. Если номер, указанный в этом параметре, уже используется будет выбран следующий доступный номер. Также обратите внимание, что если в вашей системе нет группы с идентификатором, равным этому номеру, при первом использовании **useradd** без параметра *-g* — вы получите следующее сообщение об ошибке: *useradd: unknown GID 999*, даже если учетная запись была создана правильно. Поэтому мы создали группу *users* с этим идентификатором в Раздел 7.6, «Создание основных файлов и символических ссылок».

CREATE_MAIL_SPOOL=yes

Этот параметр заставит утилиту **useradd** создавать файл почтового ящика для каждого нового пользователя. **useradd** сделает этот файл принадлежащим группе *mail* с правами доступа 0660. Если вы предпочитаете, не создавать эти файлы, выполните следующую команду:

```
sed -i '/MAIL/s/yes/no/' /etc/default/useradd
```

8.29.3. Установка пароля пользователя root

Придумайте пароль для *root* и установите командой:

```
passwd root
```

8.29.4. Содержимое пакета Shadow

Установленные программы:	chage, chfn, chgpasswd, chpasswd, chsh, expiry, faillog, getsubids, gpasswd, groupadd, groupdel, groupmems, groupmod, grpck, grpconv, grpunconv, login, logoutd, newgidmap, newgrp, newuidmap, newusers, nologin, passwd, pwck, pwconv, pwunconv, sg (ссылка на newgrp), su, useradd, userdel, usermod, vigr (ссылка на vipw) и vipw
Установленные библиотеки:	libsubid.so
Созданные каталоги:	/etc/default и /usr/include/shadow

Краткое описание

chage	Используется для изменения максимального количества дней между обязательными сменами пароля
chfn	Используется для изменения полного имени пользователя и другой информации
chgpasswd	Используется для обновления паролей групп в пакетном режиме.
chpasswd	Используется для обновления паролей пользователей в пакетном режиме.
chsh	Используется для изменения оболочки входа для пользователя.
expiry	Проверяет и применяет текущую политику истечения срока действия пароля

faillog Используется для проверки журнала неудачных попыток входа в систему, для установки максимального количества неудачных попыток до блокировки учетной записи и для сброса счетчика неудачных попыток.

getsubids Используется для перечисления подчиненных диапазонов идентификаторов для пользователя

gpasswd Используется для добавления и удаления пользователей и администраторов в группы.

groupadd Создает группу с указанным именем

groupdel Удаляет группу с указанным именем

groupmems Позволяет пользователю управлять своим собственным списком членов группы без привилегий суперпользователя

groupmod Используется для изменения имени группы или GID

grpck Проверяет целостность файлов групп `/etc/group` и `/etc/gshadow`

grpconv Создает или изменяет файл теневых групп, используя для этого обычный файл групп

grpunconv Обновляет `/etc/group` из `/etc/gshadow` , а затем удаляет последний

login Используется системой для того, чтобы пользователь мог войти в систему

logoutd Это демон, используемый для обеспечения соблюдения ограничений на время входа в систему и порты

newgidmap Используется для сопоставления gid пространства имен пользователя

newgrp Используется для изменения GID во время сеанса входа в систему

newuidmap Используется для сопоставления uid пространства имен пользователя

newusers Используется для создания или изменения последовательности учетных записей

nologin Отображает сообщение о том, что учетная запись недоступна; она предназначена для использования в качестве оболочки по умолчанию для отключенных учетных записей

passwd Используется для изменения пароля для учетной записи пользователя или группы.

pwck Проверяет целостность файлов паролей `/etc/passwd` и `/etc/shadow`

pwconv Создает или изменяет файл теневых паролей, используя для этого обычный файл паролей

pwunconv Обновляет `/etc/passwd` из `/etc/shadow` а затем удаляет последний

sg Выполняет указанную команду в случае, если у пользователя идентификатор группы GID совпадает с идентификатором указанной группы

su Запускает оболочку с заменой идентификаторов пользователя и группы

useradd Создает нового пользователя с указанным именем, либо изменяет информацию, задаваемую по умолчанию для нового пользователя

userdel Удаляет учетную запись указанного пользователя

usermod Используется для изменения имени пользователя, идентификатора пользователя (UID), оболочки, группы, домашнего каталога и т.д.

vigr Редактирует файлы `/etc/group` или `/etc/gshadow`

vipw Редактирует файлы `/etc/passwd` или `/etc/shadow`

`libsubid` библиотека для обработки подчиненных диапазонов идентификаторов пользователей и групп

8.30. GCC-14.2.0

Пакет GCC содержит коллекцию компиляторов GNU, которая включает компиляторы C и C++.

Приблизительное время сборки:	46 SBU (с тестами)
Требуемое дисковое пространство:	6.3 GB

8.30.1. Установка пакета GCC

При сборке на x86_64 измените имя каталога по умолчанию для 64-битных библиотек на «lib»:

Измените имя каталога по умолчанию для 64-битных библиотек на «lib»:

```
sed -e '/m64=/s/lib64/lib/' \
    -e '/m32=/s/m32=.*/m32=..\/lib32$(call if_multiarch,:i386-linux-gnu)/' \
    -i.orig gcc/config/i386/t-linux64
```

Документация GCC рекомендует собирать GCC в отдельном каталоге:

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

Подготовьте GCC к компиляции:

```
mlist=m64,m32
../configure --prefix=/usr               \
             LD=ld                       \
             --enable-languages=c,c++    \
             --enable-default-pie        \
             --enable-default-ssp        \
             --enable-host-pie           \
             --enable-multilib           \
             --with-multilib-list=$mlist \
             --disable-bootstrap         \
             --disable-fixincludes       \
             --with-system-zlib
```

GCC поддерживает семь различных языков программирования, но предварительные условия для большинства из них еще не выполнены. См. страницу *BLFS Book GCC* для получения инструкций о том, как собрать все языки, поддерживаемые GCC

Значение новых параметров настройки:

`LD=ld`

Этот параметр указывает скрипту configure использовать ld, установленный программой Binutils, собранной ранее в этой главе, а не кросс версию, которая использовалась бы в противном случае.

`--disable-fixincludes`

По умолчанию во время установки GCC некоторые системные заголовки будут «исправлены» для использования с GCC. Это не обязательно для современной системы Linux и потенциально опасно, если пакет будет переустановлен после установки GCC. Этот параметр не позволяет GCC «исправлять» заголовки.

`--with-system-zlib`

Этот параметр указывает GCC ссылаться на установленную в системе копию библиотеки Zlib, а не на собственную внутреннюю копию.

Примечание

PIE (позиционно-независимые исполняемые файлы) — это двоичные программы, которые можно загружать в любое место памяти. Без PIE функция безопасности под названием ASLR (рандомизация размещения адресного пространства) может применяться к общим библиотекам, но не к самим исполняемым файлам. Включение PIE позволяет использовать ASLR для исполняемых файлов в дополнение к общим библиотекам и смягчает некоторые атаки, основанные на фиксированных адресах конфиденциального кода или данных в исполняемых файлах.

SSP (Stack Smashing Protection - защита от разрушения стека) — это метод, гарантирующий, что стек параметров не будет поврежден. Повреждение стека может, например, изменить адрес возврата подпрограммы, тем самым передав управление какому-то опасному коду (существующему в программе или общих библиотеках или каким-то образом внедренному злоумышленником).

Скомпилируйте пакет:

```
make
```

Важно

В этом разделе набор тестов для GCC считается важным, но занимает много времени. Начинающим сборщикам не рекомендуется пропускать его. Время выполнения тестов можно значительно сократить, добавив -jx в приведенную ниже команду **make -k check**, где x - количество ядер процессора в вашей системе.

GCC может потребоваться больше места в стеке для компиляции некоторых сложных шаблонов кода. В качестве меры предосторожности для хост-дистрибутивов с жестким ограничением стека явно установите неограниченный жесткий предел. В большинстве хост-дистрибутивов (и в собранной LFS) жесткий предел по умолчанию неограничен, но нет ничего плохого в том, чтобы установить его явно. Также нет необходимости изменять мягкий предел размера стека, поскольку GCC автоматически установит для него соответствующее значение, если это значение не превышает жесткий предел:

```
ulimit -s -H unlimited
```

Теперь удалите/исправьте несколько известных ошибок тестирования:

```
sed -e '/cpython/d'               -i ../gcc/testsuite/gcc.dg/plugin/plugin.exp
sed -e 's/no-pic /&-no-pie /'     -i ../gcc/testsuite/gcc.target/i386/pr113689-1.c
sed -e 's/300000/(1|300000)/'     -i ../libgomp/testsuite/libgomp.c-c++-common/pr109062.c
sed -e 's/{ target nonpic } //' \
    -e '/GOTPCREL/d'              -i ../gcc/testsuite/gcc.target/i386/fentryname3.c
```

Выполните тестирование от имени непривилегированного пользователя, но не останавливайтесь на ошибках:

```
chown -R tester .
su tester -c "PATH=$PATH make -k check"
```

Чтобы получить сводку результатов набора тестов, выполните:

```
../contrib/test_summary
```

Чтобы отфильтровать только итоговую сводку, передайте вывод через pipe **`grep -A7 Summ`**.

Результаты можно сравнить с результатами, размещенными на *https://mirror.linuxfromscratch.ru/lfs/build-logs/12.2/* и *https://gcc.gnu.org/ml/gcc-testresults/*.

Не всегда удается избежать неожиданных сбоев. В некоторых случаях неудачное завершение тестов - это следствие их выполнения на конкретном оборудовании системы. Если результаты теста не сильно отличаются от результатов по указанному выше URL-адресу, можно продолжать.

Установите пакет:

```
make install
```

Каталог сборки GCC теперь принадлежит пользователю `tester`, и владелец каталога заголовочных файлов (и его содержимого) указан неверно. Измените владельца на пользователя и группу `root`:

```
chown -v -R root:root \
    /usr/lib/gcc/$(gcc -dumpmachine)/14.2.0/include{,-fixed}
```

Создайте символическую ссылку, требуемую *FHS* по "историческим" причинам.

```
ln -svr /usr/bin/cpp /usr/lib
```

Многие пакеты используют имя **cc** для вызова компилятора языка Си. Мы уже создали **cc** как символическую ссылку в GCC-Проход 2, теперь создайте символическую ссылку на его справочную страницу:

```
ln -sv gcc.1 /usr/share/man/man1/cc.1
```

Добавьте символическую ссылку совместимости, чтобы включить сборку программ с оптимизацией времени компоновки (LTO):

```
ln -sfv ../../libexec/gcc/$(gcc -dumpmachine)/14.2.0/liblto_plugin.so \
        /usr/lib/bfd-plugins/
```

Теперь, когда наш окончательный набор инструментов готов, важно еще раз убедиться, что компиляция и компоновка будут работать так, как ожидалось. Мы сделаем это, выполнив проверку работоспособности:

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

Ошибок быть не должно, и вывод последней команды будет (с учетом платформо-зависимых различий в имени динамического компоновщика):

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

Теперь убедитесь, что мы настроили использование правильных стартовых файлов:

```
grep -E -o '/usr/lib.*/S?crt[1in].*succeeded' dummy.log
```

Вывод последней команды должен быть:

```
/usr/lib/gcc/x86_64-pc-linux-gnu/14.2.0/../../../../lib/Scrt1.o succeeded
/usr/lib/gcc/x86_64-pc-linux-gnu/14.2.0/../../../../lib/crti.o succeeded
/usr/lib/gcc/x86_64-pc-linux-gnu/14.2.0/../../../../lib/crtn.o succeeded
```

В зависимости от архитектуры вашего компьютера вышеуказанные параметры могут незначительно отличаться. Разница будет заключаться в имени каталога после `/usr/lib/gcc` . Здесь важно обратить внимание на то, что **gcc** нашел все три файла `crt*.o` в каталоге `/usr/lib` .

Убедитесь, что компилятор ищет правильные заголовочные файлы:

```
grep -B4 '^ /usr/include' dummy.log
```

Эта команда должна вернуть следующий вывод:

```
#include <...> search starts here:
 /usr/lib/gcc/x86_64-pc-linux-gnu/14.2.0/include
 /usr/local/include
 /usr/lib/gcc/x86_64-pc-linux-gnu/14.2.0/include-fixed
 /usr/include
```

Опять же, имя каталога может отличаться от указанного выше, в зависимости от архитектуры вашей системы.

Затем убедитесь, что новый компоновщик использует правильные пути поиска:

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

Ссылки на пути, содержащие компоненты с '-linux-gnu', следует игнорировать, но в противном случае вывод последней команды должен быть таким:

```
SEARCH_DIR("/usr/x86_64-pc-linux-gnu/lib64")
SEARCH_DIR("/usr/local/lib64")
SEARCH_DIR("/lib64")
SEARCH_DIR("/usr/lib64")
SEARCH_DIR("/usr/x86_64-pc-linux-gnu/lib")
SEARCH_DIR("/usr/local/lib")
SEARCH_DIR("/lib")
SEARCH_DIR("/usr/lib");
```

Затем убедитесь, что мы используем правильную libc:

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

Вывод последней команды должен быть:

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

Убедитесь, что GCC использует правильный динамический компоновщик:

```
grep found dummy.log
```

Вывод последней команды должен быть (с учетом различий в имени динамического компоновщика, зависящих от платформы):

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

Если вывод выглядит не так, как показано выше, или вообще не получен, значит, где-то серьезная ошибка. Изучите и повторите шаги, чтобы выяснить, в чем проблема, и исправьте ее. Любые проблемы должны быть решены, прежде чем вы продолжите процесс.

Как только все заработает правильно, удалите тестовые файлы:

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

Наконец, переместите файл:

```
mkdir -pv /usr/share/gdb/auto-load/usr/lib
mv -v /usr/lib/*gdb.py /usr/share/gdb/auto-load/usr/lib
```

8.30.2. Содержимое пакета GCC

Установленные программы:	c++, cc (link to gcc), cpp, g++, gcc, gcc-ar, gcc-nm, gcc-ranlib, gcov, gcov-dump, gcov-tool, и lto-dump
Установленные библиотеки:	libasan.{a,so}, libatomic.{a,so}, libcc1.so, libgcc.a, libgcc_eh.a, libgcc_s.so, libgcov.a, libgomp.{a,so}, libhwasan.{a,so}, libitm.{a,so}, liblsan.{a,so}, liblto_plugin.so, libquadmath.{a,so}, libssp.{a,so}, libssp_nonshared.a, libstdc++.{a,so}, libstdc++exp.a, libstdc++fs.a, libsupc++.a, libtsan.{a,so} и libubsan.{a,so}
Созданные каталоги:	/usr/include/c++, /usr/lib/gcc, /usr/libexec/gcc и /usr/share/gcc-14.2.0

Краткое описание

c++	Компилятор C++
cc	Компилятор C

cpp	Препроцессор C; он используется компилятором для расширения инструкций #include, #define и подобные директивы в исходных файлах
g++	Компилятор C++
gcc	Компилятор C
gcc-ar	Обертка над **ar**, добавляющая плагин в командную строку. Эта программа используется только для добавления "оптимизации времени компоновки" и бесполезна с параметрами сборки по умолчанию.
gcc-nm	Обертка над **nm**, добавляющая плагин в командную строку. Эта программа используется только для добавления "оптимизации времени компоновки" и бесполезна с параметрами сборки по умолчанию.
gcc-ranlib	Обертка над **ranlib**, добавляющая плагин в командную строку. Эта программа используется только для добавления "оптимизации времени компоновки" и бесполезна с параметрами сборки по умолчанию.
gcov	Инструмент тестирования; он используется для анализа программ, чтобы определить, где оптимизация будет иметь наибольший эффект.
gcov-dump	Автономный инструмент для дампа профилей gcda and gcno
gcov-tool	Автономный инструмент обработки профиля gcda
lto-dump	Инструмент для создания дампа объектных файлов, созданных GCC с включенным LTO.
`libasan`	Библиотека времени выполнения Address Sanitizer
`libatomic`	Встроенная библиотека времени выполнения GCC atomic
`libcc1`	Библиотека, которая позволяет GDB использовать GCC
`libgcc`	Содержит средства поддержки времени исполнения для **gcc**
`libgcov`	Эта библиотека компонуется с программой, когда в GCC включено профилирование
`libgomp`	GNU реализация интерфейса OpenMP API мультиплатформенного параллельного программирования для языков C/C++ и Fortran с общим доступом к памяти
`libhwasan`	Библиотека времени выполнения Hardware-Assisted Address Sanitizer (аппаратной очистки адресов)
`libitm`	Библиотека транзакционной памяти GNU
`liblsan`	Библиотека времени выполнения Leak Sanitizer (средств защиты от утечек)
`liblto_plugin`	Плагин GCC LTO позволяет Binutils обрабатывать объектные файлы, созданные GCC с включенным LTO.
`libquadmath`	API математической библиотеки GCC Quad Precision
`libssp`	Содержит подпрограммы, поддерживающие функциональность защиты стека GCC. Обычно они не используются, потому что Glibc также предоставляет эти подпрограммы.
`libstdc++`	Стандартная библиотека C++
`libstdc++exp`	Экспериментальная библиотека контрактов C++
`libstdc++fs`	Библиотека файловой системы ISO/IEC TS 18822:2015
`libsupc++`	Предоставляет вспомогательные процедуры для языка программирования C++
`libtsan`	Библиотека времени выполнения Thread Sanitizer (средств очистки потоков)
`libubsan`	Библиотека времени выполнения Undefined Behavior Sanitizer (средств очистки неопределенного поведения)

8.31. Ncurses-6.5

Пакет Ncurses содержит библиотеки для независимой от терминала обработки ввода/вывода

Приблизительное время сборки: 0.2 SBU

Требуемое дисковое пространство: 46 MB

8.31.1. Установка пакета Ncurses

Подготовьте Ncurses к компиляции:

```
./configure --prefix=/usr           \
            --mandir=/usr/share/man \
            --with-shared           \
            --without-debug         \
            --without-normal        \
            --with-cxx-shared       \
            --enable-pc-files       \
            --with-pkg-config-libdir=/usr/lib/pkgconfig
```

Значение новых параметров настройки:

`--with-shared`
Этот параметр позволяет Ncurses собирать и устанавливать общие библиотеки C.

`--without-normal`
Этот параметр отключает сборку и установку большинства статических библиотек C.

`--without-debug`
Этот параметр предотвращает сборку и установку отладочных библиотек.

`--with-cxx-shared`
Это аргумент позволяет Ncurses собирать и устанавливать общие привязки C++. А также предотвращает сборку и установку статических привязок C++.

`--enable-pc-files`
Этот параметр генерирует и устанавливает файлы .pc для pkg-config.

Скомпилируйте пакет:

```
make
```

У этого пакета есть набор тестов, но его можно запустить только после того, как пакет будет установлен. Тесты находятся в каталоге `test/`. Дополнительные сведения смотрите в файле `README` в этом каталоге.

Установка этого пакета приведет к перезаписи `libncursesw.so.6.5` . Это может привести к сбою процесса оболочки, который использует код и данные из файла библиотеки. Установите пакет с помощью `DESTDIR` и правильно замените файл библиотеки с помощью команды **install**(также отредактирован заголовочный файл `curses.h` , чтобы обеспечить использование ABI расширенного набора символов, как это сделано в Раздел 6.3, «Ncurses-6.5»).

```
make DESTDIR=$PWD/dest install
install -vm755 dest/usr/lib/libncursesw.so.6.5 /usr/lib
rm -v  dest/usr/lib/libncursesw.so.6.5
sed -e 's/^#if.*XOPEN.*$/#if 1/' \
    -i dest/usr/include/curses.h
cp -av dest/* /
```

Многие приложения ожидают, что компоновщик может найти библиотеки Ncurses, не поддерживающие расширенный набор символов. Используя трюк ниже, свяжите такие приложения с библиотеками расширенного набора символов с помощью символических ссылок (обратите внимание, что ссылки `.so` безопасны только в том случае, если `curses.h` отредактирован так, чтобы всегда использовать ABI расширенного набора символов):

```
for lib in ncurses form panel menu ; do
    ln -sfv lib${lib}w.so /usr/lib/lib${lib}.so
    ln -sfv ${lib}w.pc    /usr/lib/pkgconfig/${lib}.pc
done
```

Убедитесь, что старые приложения, которым нужна `-lcurses` для сборки, собираются правильно:

```
ln -sfv libncursesw.so /usr/lib/libcurses.so
```

По желанию установите документацию Ncurses:

```
cp -v -R doc -T /usr/share/doc/ncurses-6.5
```

Примечание

С помощью приведенных выше инструкций не создаются библиотеки Ncurses, не использующие расширенный набор символов, поскольку ни один пакет, установленный путем компиляции из исходного кода, не будет связан с ними во время выполнения. Тем не менее, известно что некоторые бинарные приложения, которые связаны с библиотекой Ncurses и не поддерживающие расширенный набор символов, требуют наличия версии 5. Если вам необходимо иметь такие библиотеки из-за какого-либо приложения, имеющегося только в бинарном виде, или для обеспечения совместимости с LSB, соберите пакет с помощью следующих команд:

```
make distclean
./configure --prefix=/usr    \
            --with-shared    \
            --without-normal \
            --with-cxx-shared \
            --without-debug  \
            --without-cxx-binding \
            --with-abi-version=5
make sources libs
cp -av lib/lib*.so.5* /usr/lib
```

8.31.2. Сборка Ncurses - 32-битная версия

Выполните очистку после предыдущей сборки:

```
make distclean
```

Подготовьте Ncurses к компиляции:

```
CC="gcc -m32" CXX="g++ -m32" \
./configure --prefix=/usr           \
            --host=i686-pc-linux-gnu \
            --libdir=/usr/lib32     \
            --mandir=/usr/share/man \
            --with-shared           \
            --without-debug         \
            --without-normal        \
            --with-cxx-shared       \
            --enable-pc-files       \
            --with-pkg-config-libdir=/usr/lib32/pkgconfig
```

Скомпилируйте пакет:

```
make
```

Установите пакет:

```
make DESTDIR=$PWD/DESTDIR install
mkdir -p DESTDIR/usr/lib32/pkgconfig
for lib in ncurses form panel menu ; do
    rm -vf                    DESTDIR/usr/lib32/lib${lib}.so
    echo "INPUT(-l${lib}w)" > DESTDIR/usr/lib32/lib${lib}.so
    ln -svf ${lib}w.pc        DESTDIR/usr/lib32/pkgconfig/$lib.pc
done
rm -vf                     DESTDIR/usr/lib32/libcursesw.so
echo "INPUT(-lncursesw)" > DESTDIR/usr/lib32/libcursesw.so
ln -sfv libncurses.so      DESTDIR/usr/lib32/libcurses.so
cp -Rv DESTDIR/usr/lib32/* /usr/lib32
rm -rf DESTDIR
```

8.31.3. Содержимое пакета Ncurses

Установленные программы:	captoinfo (ссылка на tic), clear, infocmp, infotocap (ссылка на tic), ncursesw6-config, reset (ссылка на tset), tabs, tic, toe, tput и tset
Установленные библиотеки:	libcurses.so (символическая ссылка), libform.so (символическая ссылка), libformw.so, libmenu.so (символическая ссылка), libmenuw.so, libncurses.so (символическая ссылка), libncursesw.so, libncurses++w.so, libpanel.so (символическая ссылка) и libpanelw.so,
Созданные каталоги:	/usr/share/tabset, /usr/share/terminfo и /usr/share/doc/ncurses-6.5

Краткое описание

captoinfo	Преобразует описание termcap в описание terminfo
clear	Очищает экран, если это возможно
infocmp	Сравнивает или показывает описания terminfo
infotocap	Преобразует описание terminfo в описание termcap
ncursesw6-config	Предоставляет информацию о конфигурации пакету ncurses
reset	Повторно инициализирует терминал со значениями по умолчанию
tabs	Очищает и устанавливает размеры табуляции в терминале
tic	Компилятор описания terminfo, преобразует файл terminfo из исходного формата в двоичный, который необходим для подпрограмм библиотеки ncurses [Файл terminfo содержит информацию о возможностях конкретного терминала.]
toe	Выводит список всех доступных типов терминалов, для каждого из которых указывается его имя и приводится описание
tput	Позволяет использовать в командной оболочке настройки, относящиеся к особенностям конкретного терминала; может также использоваться для сброса или инициализации терминала, либо для вывода полного имени терминала
tset	Может использоваться для инициализации терминалов
`libncursesw`	Содержит функции, отображающие различными способами текст на экране терминала. Хорошим примером использования этих функций является меню, отображаемое командой **make menuconfig** при настройке ядра
`libncurses++w`	Содержит функции связывания C++ с другими библиотеками в пакете
`libformw`	Содержит функции, реализующие формы
`libmenuw`	Содержит функции, реализующие меню
`libpanelw`	Содержит функции, реализующие панели

8.32. Sed-4.9

Пакет Sed содержит потоковый редактор текста

Приблизительное время сборки: 0.3 SBU
Требуемое дисковое пространство: 30 MB

8.32.1. Установка пакета Sed

Подготовьте Sed к компиляции:

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

Скомпилируйте пакет и сгенерируйте HTML-документацию:

```
make
make html
```

Чтобы протестировать пакет, выполните:

```
chown -R tester .
su tester -c "PATH=$PATH make check"
```

Установите пакет и документацию к нему:

```
make install
install -d -m755           /usr/share/doc/sed-4.9
install -m644 doc/sed.html /usr/share/doc/sed-4.9
```

8.32.2. Содержимое пакета Sed

Установленные программы: sed
Созданные каталоги: /usr/share/doc/sed-4.9

Краткое описание

sed Фильтрует и преобразует текстовые файлы за один проход

8.33. Psmisc-23.7

Пакет Psmisc содержит программы для отображения информации о запущенных процессах.

Приблизительное время сборки:	менее 0.1 SBU
Требуемое дисковое пространство:	6.7 MB

8.33.1. Установка пакета Psmisc

Подготовьте Psmisc к компиляции:

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

Скомпилируйте пакет:

```
make
```

Чтобы протестировать пакет, выполните:

```
make check
```

Установите пакет:

```
make install
```

8.33.2. Содержимое пакета Psmisc

Установленные программы:	fuser, killall, peekfd, prtstat, pslog, pstree и pstree.x11 (ссылка на pstree)

Краткое описание

fuser	Сообщает идентификаторы процессов (PID), которые используют данные файлы или файловые системы.
killall	Уничтожает процессы по имени; посылает сигнал всем процессам, выполняющим любую из заданных команд
peekfd	Просматривает файловые дескрипторы запущенного процесса с учетом его PID
prtstat	Выводит информацию о процессе
pslog	Сообщает текущий путь к журналам процесса
pstree	Отображает запущенные процессы в виде дерева
pstree.x11	То же, что и **pstree**, за исключением того, что он ожидает подтверждения перед выходом.

8.34. Gettext-0.22.5

Пакет Gettext содержит утилиты для интернационализации и локализации. Они позволяют компилировать программы с поддержкой NLS (Native Language Support), позволяя им выводить сообщения на родном языке пользователя.

Приблизительное время сборки: 1.6 SBU
Требуемое дисковое пространство: 260 MB

8.34.1. Установка пакета Gettext

Подготовьте Gettext для компиляции:

```
./configure --prefix=/usr    \
            --disable-static \
            --docdir=/usr/share/doc/gettext-0.22.5
```

Скомпилируйте пакет:

```
make
```

Чтобы протестировать пакет (это занимает много времени, около 3 SBU), выполните:

```
make check
```

Установите пакет:

```
make install
chmod -v 0755 /usr/lib/preloadable_libintl.so
```

8.34.2. Содержимое пакета Gettext

Установленные программы: autopoint, envsubst, gettext, gettext.sh, gettextize, msgattrib, msgcat, msgcmp, msgcomm, msgconv, msgen, msgexec, msgfilter, msgfmt, msggrep, msginit, msgmerge, msgunfmt, msguniq, ngettext, recode-sr-latin, и xgettext
Установленные библиотеки: libasprintf.so, libgettextlib.so, libgettextpo.so, libgettextsrc.so, libtextstyle.so и preloadable_libintl.so
Созданные каталоги: /usr/lib/gettext, /usr/share/doc/gettext-0.22.5, /usr/share/gettext и /usr/share/gettext-0.22.5

Краткое описание

autopoint Копирует файлы стандартной инфраструктуры Gettext в пакет с исходным кодом

envsubst Подставляет переменные окружения в строки, используемые командной оболочкой

gettext Переводит сообщение с естественного языка на язык пользователя, осуществляя для этого поиск уже сделанного перевода в каталоге сообщений

gettext.sh В основном служит библиотекой функций оболочки для gettext.

gettextize Копирует все стандартные файлы Gettext в указанный каталог верхнего уровня пакета, чтобы начать его интернационализацию.

msgattrib Фильтрует сообщения каталога переводов в соответствии с их атрибутами и управляет атрибутами

msgcat Объединяет указанные файлы `.po`

msgcmp	Сравнивает два файла `.po` , чтобы проверить, что оба содержат один и тот же набор строк msgid
msgcomm	Находит сообщения, которые являются общими для указанных файлов `.po`
msgconv	Преобразует каталог переводов в другую кодировку символов
msgen	Создает каталог переводов на английский язык
msgexec	Применяет команду ко всем переводам каталога переводов
msgfilter	Применяет фильтр ко всем переводам каталога переводов
msgfmt	Генерирует каталог двоичных сообщений из каталога переводов
msggrep	Извлекает все сообщения каталога переводов, которые соответствуют заданному шаблону или принадлежат нескольким указанным исходным файлам
msginit	Создает новый файл `.po` , инициализируя метаинформацию значениями из среды пользователя.
msgmerge	Объединяет два необработанных перевода в один файл
msgunfmt	Декомпилирует каталог двоичных сообщений в необработанный текст перевода
msguniq	Объединяет дублирующиеся переводы в каталоге переводов
ngettext	Отображает перевод текстового сообщения на родной язык, грамматическая форма которого зависит от числа.
recode-sr-latin	Перекодирует сербский текст с кириллицы на латиницу.
xgettext	Извлекает переводимые строки сообщений из заданных исходных файлов для создания первого шаблона перевода.
`libasprintf`	Определяет класс *autosprintf*, который делает подпрограммы вывода в формате C пригодными для использования в программах на C++ для использования со строками <*string*> и потоками <*iostream*>
`libgettextlib`	Содержит общие подпрограммы, используемые различными программами Gettext; они не предназначены для общего использования
`libgettextpo`	Используется для написания специализированных программ, обрабатывающих файлы `.po` ; эта библиотека используется, когда стандартных приложений, поставляемых с Gettext (таких как **msgcomm**, **msgcmp**, **msgattrib**, и **msgen**), недостаточно.
`libgettextsrc`	Предоставляет общие подпрограммы, используемые различными программами Gettext; они не предназначены для общего использования
`libtextstyle`	Библиотека стилей текста
`preloadable_libintl`	Библиотека, предназначенная для использования LD_PRELOAD, которая помогает `libintl` записывать в журнал непереведённые сообщения

8.35. Bison-3.8.2

Пакет Bison содержит генератор синтаксического анализа.

Приблизительное время сборки:	2.2 SBU
Требуемое дисковое пространство:	62 MB

8.35.1. Установка пакета Bison

Подготовьте Bison к компиляции:

```
./configure --prefix=/usr --docdir=/usr/share/doc/bison-3.8.2
```

Скомпилируйте пакет:

```
make
```

Для тестирования пакета (около 5,5 SBU), выполните:

```
make check
```

Установите пакет:

```
make install
```

8.35.2. Содержимое пакета Bison

Установленные программы:	bison и yacc
Установленные библиотеки:	liby.a
Созданные каталоги:	/usr/share/bison

Краткое описание

bison Генерирует из набора правил программу для анализа структуры текстовых файлов; Bison является заменой Yacc (Yet Another Compiler Compiler)

yacc Обертка для **bison**, предназначенная для программ, которые до сих пор вызывают **yacc** вместо **bison**; он вызывает **bison** с параметром *-y*

`liby` Библиотека Yacc, содержащая реализации Yacc-совместимых функций `yyerror` и `main`; обычно эта библиотека не очень нужна, но требуется POSIX

8.36. Grep-3.11

Пакет Grep содержит программы для поиска по содержимому файлов.

Приблизительное время сборки: 0.4 SBU
Требуемое дисковое пространство: 39 MB

8.36.1. Установка пакета Grep

Во-первых, удалите предупреждение об использовании egrep и fgrep, которое приводит к сбою тестов некоторых пакетов:

```
sed -i "s/echo/#echo/" src/egrep.sh
```

Подготовьте Grep к компиляции:

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

Скомпилируйте пакет:

```
make
```

Чтобы протестировать пакет, выполните:

```
make check
```

Установите пакет:

```
make install
```

8.36.2. Содержимое пакета Grep

Установленные программы: egrep, fgrep, и grep

Краткое описание

egrep Выводит строки, соответствующие расширенному регулярному выражению. Команда устарела, вместо неё используйте **grep -E**

fgrep Выводит строки, соответствующие списку фиксированных строк. Команда устарела, вместо неё используйте **grep -F**

grep Выводит строки, соответствующие простому регулярному выражению

8.37. Bash-5.2.32

Пакет Bash содержит Bourne-Again Shell.

Приблизительное время сборки:	1.2 SBU
Требуемое дисковое пространство:	52 MB

8.37.1. Установка пакета Bash

Подготовьте Bash к компиляции:

```
./configure --prefix=/usr             \
            --without-bash-malloc     \
            --with-installed-readline \
            bash_cv_strtold_broken=no \
            --docdir=/usr/share/doc/bash-5.2.32
```

Значение нового параметра настройки:

`--with-installed-readline`
: Этот параметр указывает Bash использовать библиотеку `readline`, которая уже установлена в системе, а не собственную версию readline.

Скомпилируйте пакет:

```
make
```

Перейдите к разделу «Установка пакета», если не планируете запускать тесты.

Перед запуском тестов, убедитесь, что пользователь `tester` может писать в каталог с исходниками:

```
chown -R tester .
```

Набор тестов пакета предназначен для запуска пользователем без полномочий root, которому принадлежит терминал, подключенный к стандартному вводу. Чтобы удовлетворить это требование, создайте новый псевдотерминал с помощью Expect и запустите тесты от имени пользователя `tester`:

```
su -s /usr/bin/expect tester << "EOF"
set timeout -1
spawn make tests
expect eof
lassign [wait] _ _ _ value
exit $value
EOF
```

Набор тестов использует **diff** для определения разницы между выводом тестового сценария и ожидаемым результатом. Любой вывод **diff** (с префиксом < и >) указывает на сбой теста, если нет сообщение о том, что различия можно проигнорировать. Известно, что один тест с именем `run-builtins` не работает на некоторых хост-дистрибутивах, указывая на различия в первой строке выходных данных.

Установите пакет:

```
make install
```

Запустите только что скомпилированную программу **bash** (заменив ту, которая выполняется в данный момент):

```
exec /usr/bin/bash --login
```

8.37.2. Содержимое пакета Bash

Установленные программы: bash, bashbug и sh (ссылка на bash)
Созданные каталоги: /usr/include/bash, /usr/lib/bash, and /usr/share/doc/bash-5.2.32

Краткое описание

bash Широко распространенный командный интерпретатор. Выполняет различные дополнения и подстановки в переданной командной строке перед её выполнением, что делает этот интерпретатор мощным инструментом

bashbug Скрипт, помогающий пользователю составлять и отправлять по почте отчеты об ошибках **bash**

sh Симлинк на программу **bash**; при вызове **sh**, **bash** пытается максимально точно имитировать поведение **sh**, при этом также соответствуя стандарту POSIX.

8.38. Libtool-2.4.7

Пакет Libtool содержит сценарий поддержки универсальной библиотеки GNU. Это упрощает использование общих библиотек благодаря согласованному переносимому интерфейсу.

Приблизительное время сборки: 0.8 SBU

Требуемое дисковое пространство: 45 MB

8.38.1. Установка пакета Libtool

Подготовьте Libtool к компиляции:

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

Скомпилируйте пакет:

```
make
```

Чтобы протестировать пакет, выполните:

```
make -k check
```

Известно, что пять тестов в среде сборки LFS завершаются неудачно из-за циклической зависимости, но эти тесты проходят успешно, если запустить их повторно после установки automake. Кроме того, в grep-3.8 или более поздней версии два теста вызовут предупреждение для регулярных выражений, несовместимых с POSIX и завершатся ошибкой.

Установите пакет:

```
make install
```

Удалите ненужную статическую библиотеку:

```
rm -fv /usr/lib/libltdl.a
```

8.38.2. Установка Libtool - 32-битная версия

Выполните очистку после предыдущей сборки:

```
make distclean
```

Подготовьте Libtool к компиляции:

```
CC="gcc -m32" ./configure \
    --host=i686-pc-linux-gnu \
    --prefix=/usr            \
    --libdir=/usr/lib32
```

Скомпилируйте пакет:

```
make
```

Установите пакет:

```
make DESTDIR=$PWD/DESTDIR install
cp -Rv DESTDIR/usr/lib32/* /usr/lib32
rm -rf DESTDIR
```

8.38.3. Содержимое пакета Libtool

Установленные программы: libtool и libtoolize
Установленные библиотеки: libltdl.so
Созданные каталоги: /usr/include/libltdl и /usr/share/libtool

Краткое описание

libtool Обеспечивает общие услуги поддержки при сборке библиотек

libtoolize Предоставляет стандартный способ добавления поддержки **libtool** в пакет

`libltdl` Скрывает различные проблемы, связанные с открытием динамически загружаемых библиотек

8.39. GDBM-1.24

Пакет GDBM содержит менеджер баз данных GNU. Это библиотека функций базы данных, использующая расширяемое хеширование и работающая аналогично стандартной СУБД UNIX. Библиотека предоставляет примитивы для хранения пар ключ/значение, поиска и извлечения данных по его ключу и удаления ключа вместе с его данными.

Приблизительное время сборки: менее 0.1 SBU
Требуемое дисковое пространство: 13 MB

8.39.1. Установка пакета GDBM

Подготовьте GDBM к компиляции:

```
./configure --prefix=/usr    \
            --disable-static \
            --enable-libgdbm-compat
```

Значение параметра configure:

`--enable-libgdbm-compat`
Этот параметр включает сборку библиотеки совместимости libgdbm. Некоторым пакетам за пределами LFS могут потребоваться более старые подпрограммы DBM, которые он предоставляет.

Скомпилируйте пакет:

```
make
```

Чтобы протестировать пакет, выполните:

```
make check
```

Установите пакет:

```
make install
```

8.39.2. Установка GDBM - 32-битная версия

Выполните очистку после предыдущей сборки:

```
make distclean
```

Подготовьте GDBM к компиляции:

```
CC="gcc -m32" CXX="g++ -m32" ./configure \
    --host=i686-pc-linux-gnu      \
    --prefix=/usr                 \
    --libdir=/usr/lib32           \
    --disable-static              \
    --enable-libgdbm-compat
```

Скомпилируйте пакет:

```
make
```

Установите пакет:

```
make DESTDIR=$PWD/DESTDIR install
cp -Rv DESTDIR/usr/lib32/* /usr/lib32/
rm -rf DESTDIR
```

8.39.3. Содержимое пакета GDBM

Установленные программы:	gdbm_dump, gdbm_load, и gdbmtool
Установленные библиотеки:	libgdbm.so и libgdbm_compat.so

Краткое описание

gdbm_dump	Сохраняет дамп базы данных GDBM в файл
gdbm_load	Восстанавливает базу данных GDBM из дампа.
gdbmtool	Проверяет и изменяет базу данных GDBM
`libgdbm`	Содержит функции для управления хэшированной базой данных
`libgdbm_compat`	Библиотека совместимости, содержащая более старые функции DBM

8.40. Gperf-3.1

Gperf генерирует идеальную хэш-функцию из набора ключей.

Приблизительное время сборки: менее 0.1 SBU
Требуемое дисковое пространство: 6.1 MB

8.40.1. Установка пакета Gperf

Подготовьте Gperf к компиляции:

```
./configure --prefix=/usr --docdir=/usr/share/doc/gperf-3.1
```

Скомпилируйте пакет:

```
make
```

Известно, что тесты завершаются ошибкой при одновременном выполнении нескольких тестов (параметр -j больше 1). Для выполнения тестов, запустите следующую команду:

```
make -j1 check
```

Установите пакет:

```
make install
```

8.40.2. Содержимое пакета Gperf

Установленные программы: gperf
Созданные каталоги: /usr/share/doc/gperf-3.1

Краткое описание

gperf Генерирует идеальный хэш из набора ключей

8.41. Expat-2.6.2

Пакет Expat содержит потоковую библиотеку C для синтаксического анализа XML

Приблизительное время сборки: 0.1 SBU
Требуемое дисковое пространство: 13 MB

8.41.1. Установка пакета Expat

Подготовьте Expat к компиляции:

```
./configure --prefix=/usr    \
            --disable-static \
            --docdir=/usr/share/doc/expat-2.6.2
```

Скомпилируйте пакет:

```
make
```

Чтобы протестировать пакет, выполните:

```
make check
```

Установите пакет:

```
make install
```

По желанию установите документацию:

```
install -v -m644 doc/*.{html,css} /usr/share/doc/expat-2.6.2
```

8.41.2. Установка Expat - 32-битная версия

Выполните очистку после предыдущей сборки, но сохраните скомпилированную документацию, поскольку её невозможно пересобрать на этом этапе:

```
sed -e "/^am__append_1/ s/doc//" -i Makefile
make clean
```

Подготовьте Expat к компиляции:

```
CC="gcc -m32" ./configure \
    --prefix=/usr        \
    --disable-static     \
    --libdir=/usr/lib32  \
    --host=i686-pc-linux-gnu
```

Скомпилируйте пакет:

```
make
```

Установите пакет:

```
make DESTDIR=$PWD/DESTDIR install
cp -Rv DESTDIR/usr/lib32/* /usr/lib32
rm -rf DESTDIR
```

8.41.3. Содержимое пакета Expat

Установленные программы: xmlwf
Установленные библиотеки: libexpat.so
Созданные каталоги: /usr/share/doc/expat-2.6.2

Краткое описание

xmlwf	Утилита проверки правильности формирования XML документов
`libexpat`	Содержит функции API для синтаксического анализа XML

8.42. Inetutils-2.5

Пакет Inetutils содержит базовые программы для работы с сетью.

Приблизительное время сборки: 0.2 SBU
Требуемое дисковое пространство: 35 MB

8.42.1. Установка пакета Inetutils

Сначала внесите некоторые изменения, чтобы сборку пакета с помощью gcc-14.1 или более поздней версии:

```
sed -i 's/def HAVE_TERMCAP_TGETENT/ 1/' telnet/telnet.c
```

Подготовьте Inetutils к компиляции:

```
./configure --prefix=/usr        \
            --bindir=/usr/bin    \
            --localstatedir=/var \
            --disable-logger     \
            --disable-whois      \
            --disable-rcp        \
            --disable-rexec      \
            --disable-rlogin     \
            --disable-rsh        \
            --disable-servers
```

Значение параметров настройки:

`--disable-logger`
Параметр запрещает установку программы **logger**, используемой скриптами для отправки сообщений системной службе логирования (System Log Daemon). Не устанавливайте её, т.к. Util-linux устанавливает более свежую версию.

`--disable-whois`
Этот параметр отключает сборку **whois**-клиента Inetutils, который устарел. Инструкции для сборки более нового клиента **whois** находятся в книге BLFS.

`--disable-r*`
Отключает установку устаревших программ, которые не должны использоваться по соображениям безопасности. Функционал этих программы можно получить установкой пакета openssh из книги BLFS.

`--disable-servers`
Отключает установку различных сетевых серверов, входящих в состав пакета Inetutils. Эти серверы считаются неподходящими для базовой системы LFS. Некоторые из них небезопасны по своей природе и считаются надежными только в доверенных сетях. Обратите внимание, что для многих из них доступны более качественные замены.

Скомпилируйте пакет:

```
make
```

Чтобы протестировать пакет, выполните:

```
make check
```

Установите пакет:

```
make install
```

Переместите программу в правильное место:

```
mv -v /usr/{,s}bin/ifconfig
```

8.42.2. Содержимое пакета Inetutils

Установленные программы: dnsdomainname, ftp, ifconfig, hostname, ping, ping6, talk, telnet, tftp и traceroute

Краткое описание

dnsdomainname Показывает системное DNS имя

ftp Программа для передачи файлов по протоколу FTP

hostname Сообщает или задает имя хоста

ifconfig Управляет сетевыми интерфейсами

ping Отправляет пакеты эхо-запросов и сообщает, сколько времени занимают ответы

ping6 Версия **ping** для сетей IPv6

talk Используется для общения с другими пользователями

telnet Интерфейс к протоколу TELNET

tftp Программа для передачи файлов по протоколу TFTP (Trivial File Transfer Protocol — простой протокол передачи файлов)

traceroute Отслеживает маршрут, по которому проходят ваши пакеты от хоста на которым вы работаете, к другому узлу сети, показывая все промежуточные переходы (шлюзы) на этом пути.

8.43. Less-661

Пакет Less содержит средство просмотра текстовых файлов

Приблизительное время сборки: менее 0.1 SBU
Требуемое дисковое пространство: 14 MB

8.43.1. Установка пакета Less

Подготовьте Less к компиляции:

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

Значение параметров настройки:

`--sysconfdir=/etc`
Этот параметр указывает программам, созданным пакетом, искать файлы конфигурации в `/etc` .

Скомпилируйте пакет:

```
make
```

Чтобы протестировать пакет, выполните:

```
make check
```

Установите пакет:

```
make install
```

8.43.2. Содержимое пакета Less

Установленные программы: less, lessecho и lesskey

Краткое описание

less Просмотрщик файлов; отображает содержимое данного файла, позволяя пользователю прокручивать файл, искать строки и переходить к меткам

lessecho Требуется для расширения метасимволов, таких как * и *?*, в именах файлов в системах Unix

lesskey Используется для привязки клавиш в программе **less**

8.44. Perl-5.40.0

Пакет Perl содержит практический язык для извлечения данных и составления отчётов (Practical Extraction and Report Language).

Приблизительное время сборки: 1.4 SBU
Требуемое дисковое пространство: 245 MB

8.44.1. Установка пакета Perl

Эта версия Perl собирает модули Compress::Raw::Zlib и Compress::Raw::BZip2. По умолчанию Perl будет использовать внутреннюю копию исходников для сборки. Выполните следующую команду, чтобы Perl использовал библиотеки, установленные в системе:

```
export BUILD_ZLIB=False
export BUILD_BZIP2=0
```

Чтобы иметь полный контроль над настройкой Perl, вы можете удалить параметры «-des» из следующей команды и вручную выбрать способ сборки этого пакета. В качестве альтернативы, используйте команду точно так, как указано ниже, чтобы использовать значения по умолчанию, которые Perl определяет автоматически:

```
sh Configure -des                                         \
             -D prefix=/usr                               \
             -D vendorprefix=/usr                         \
             -D privlib=/usr/lib/perl5/5.40/core_perl     \
             -D archlib=/usr/lib/perl5/5.40/core_perl     \
             -D sitelib=/usr/lib/perl5/5.40/site_perl     \
             -D sitearch=/usr/lib/perl5/5.40/site_perl    \
             -D vendorlib=/usr/lib/perl5/5.40/vendor_perl \
             -D vendorarch=/usr/lib/perl5/5.40/vendor_perl \
             -D man1dir=/usr/share/man/man1               \
             -D man3dir=/usr/share/man/man3               \
             -D pager="/usr/bin/less -isR"                \
             -D useshrplib                                \
             -D usethreads
```

Значение параметров Configure:

`-D pager="/usr/bin/less -isR"`
Параметр указывает использовать **`less`** вместо **`more`**.

`-D man1dir=/usr/share/man/man1 -D man3dir=/usr/share/man/man3`
Так как Groff еще не установлен, **Configure** не будет создавать man-страницы для Perl. Эти параметры переопределяют это поведение.

`-D usethreads`
Собрать Perl с поддержкой потоков.

Скомпилируйте пакет:

```
make
```

Чтобы протестировать пакет (примерно 11 SBU), выполните:

```
TEST_JOBS=$(nproc) make test_harness
```

Установка пакета и очистка:

```
make install
unset BUILD_ZLIB BUILD_BZIP2
```

8.44.2. Содержимое пакета Perl

Установленные программы:	corelist, cpan, enc2xs, encguess, h2ph, h2xs, instmodsh, json_pp, libnetcfg, perl, perl5.40.0 (жесткая ссылка на perl), perlbug, perldoc, perlivp, perlthanks (жесткая ссылка на perlbug), piconv, pl2pm, pod2html, pod2man, pod2text, pod2usage, podchecker, podselect, prove, ptar, ptardiff, ptargrep, shasum, splain, xsubpp и zipdetails
Установленные библиотеки:	Список слишком большой для перечисления
Созданные каталоги:	/usr/lib/perl5

Краткое описание

corelist	Интерфейс командной строки для Module::CoreList
cpan	Позволяет получать из командной строки доступ к архиву документации и программ Perl (Comprehensive Perl Archive Network - CPAN)
enc2xs	Собирает расширение Perl для модуля Encode либо с использование таблицы символов Unicode, либо с использованием файлов кодирования Tcl
encguess	Определяет тип кодировки одного или нескольких файлов
h2ph	Конвертирует заголовочные файлы Си `.h` в заголовочные файлы Perl `.ph`
h2xs	Конвертирует заголовочные файлы Си `.h` в расширения Perl
instmodsh	Сценарий оболочки для проверки установленных модулей Perl; он может создать архив из установленного модуля.
json_pp	Преобразует данные между определенными входными и выходными форматами
libnetcfg	Может использоваться для настройки Perl-модуля `libnet`
perl	Объединяет лучшие возможности C, **sed**, **awk** и **sh** в одном языке
perl5.40.0	Жесткая ссылка на **perl**
perlbug	Используется для создания отчетов об ошибках в Perl или модулях, которые поставляются с ним, и отправки их по почте
perldoc	Отображает часть документации в формате pod, которая встроена в дерево установки Perl или в сценарий Perl
perlivp	Процедура проверки установки Perl; ее можно использовать для проверки правильности установки Perl и его библиотек
perlthanks	Используется для создания сообщения-благодарности, отсылаемого разработчикам Perl
piconv	Perl версия конвертера **iconv**, используемого для кодирования символов
pl2pm	Инструмент для грубого конвертирования файлов `.pl` Perl4 в модули `.pm` Perl5
pod2html	Преобразует файлы из формата pod в формат HTML
pod2man	Преобразует данные pod в форматированный входной поток для *roff
pod2text	Преобразует данные pod в форматированный текст ASCII
pod2usage	Печатает в файл сообщения usage из встроенных документов pod
podchecker	Проверяет синтаксис файлов документации формата pod
podselect	Отображает выбранные разделы документации pod
prove	Инструмент командной строки для выполнения тестов с помощью модуля Test::Harness
ptar	Программа, похожая на **tar**, написанная на Perl

ptardiff	Программа на Perl для сравнения распакованного и нераспакованного архивов
ptargrep	Программа на Perl для текстового поиска по шаблону внутри tar-архива
shasum	Печатает или проверяет контрольные суммы SHA
splain	Включает подробные предупреждения для диагностики в Perl
xsubpp	Преобразует код Perl XS в код C
zipdetails	Отображает сведения о внутренней структуре Zip-файла

8.45. XML::Parser-2.47

Модуль XML::Parser представляет собой Perl-интерфейс к XML-парсеру Джеймса Кларка Expat.

Приблизительное время сборки: менее 0.1 SBU
Требуемое дисковое пространство: 2.4 MB

8.45.1. Установка пакета XML::Parser

Подготовьте XML::Parser к компиляции:

```
perl Makefile.PL
```

Скомпилируйте пакет:

```
make
```

Чтобы протестировать пакет, выполните:

```
make test
```

Установите пакет:

```
make install
```

8.45.2. Содержимое XML::Parser

Установленный модуль: Expat.so

Краткое описание

`Expat` предоставляет Perl интерфейс для Expat

8.46. Intltool-0.51.0

Intltool — это инструмент интернационализации, используемый для извлечения переводимых строк из исходных файлов.

Приблизительное время сборки: менее 0.1 SBU
Требуемое дисковое пространство: 1.5 MB

8.46.1. Установка пакета Intltool

Сначала исправьте предупреждение, вызываемое perl-5.22 и более поздними версиями:

```
sed -i 's:\\\${:\\\$\\{:' intltool-update.in
```

Примечание

Приведенное выше регулярное выражение выглядит необычно из-за множества слэшей. Что оно делает, так это добавляет обратную косую черту перед правой фигурной скобкой в последовательности '\${' в результате чего получается '\$\{'.

Подготовьте Intltool к компиляции:

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

Скомпилируйте пакет:

```
make
```

Чтобы протестировать пакет, выполните:

```
make check
```

Установите пакет:

```
make install
install -v -Dm644 doc/I18N-HOWTO /usr/share/doc/intltool-0.51.0/I18N-HOWTO
```

8.46.2. Содержимое пакета Intltool

Установленные программы: intltool-extract, intltool-merge, intltool-prepare, intltool-update и intltoolize
Созданные каталоги: /usr/share/doc/intltool-0.51.0 и /usr/share/intltool

Краткое описание

intltoolize Подготавливает пакет для использования intltool

intltool-extract Генерирует заголовочные файлы, которые могут быть прочитаны с помощью **gettext**

intltool-merge Объединяет переведенные строки в файлы различных типов

intltool-prepare Обновляет файлы pot и объединяет их с файлами перевода

intltool-update Обновляет файлы шаблонов po и объединяет их с переводами

8.47. Autoconf-2.72

Пакет Autoconf содержит программы для создания сценариев оболочки, которые могут автоматически настраивать исходный код.

Приблизительное время сборки: менее 0.1 SBU (около 1.8 SBU с тестами)
Требуемое дисковое пространство: 25 MB

8.47.1. Установка пакета Autoconf

Подготовьте Autoconf к компиляции:

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

Скомпилируйте пакет:

```
make
```

Чтобы протестировать пакет, выполните:

```
make check
```

Установите пакет:

```
make install
```

8.47.2. Содержимое пакета Autoconf

Установленные программы: autoconf, autoheader, autom4te, autoreconf, autoscan, autoupdate, и ifnames
Созданные каталоги: /usr/share/autoconf

Краткое описание

autoconf Генерирует сценарии оболочки, которые автоматически настраивают пакеты исходного кода программного обеспечения для адаптации ко многим типам Unix-подобных систем; создаваемые сценарии независимы—для их запуска не требуется программа **autoconf**.

autoheader Инструмент для создания файлов шаблонов операторов C *#define* для использования в configure

autom4te Обертка для макропроцессора M4

autoreconf Автоматически запускает **autoconf**, **autoheader**, **aclocal**, **automake**, **gettextize** и **libtoolize** в правильном порядке, чтобы сэкономить время, при внесении изменений в файлы шаблонов **autoconf** и **automake**.

autoscan Помогает создать файл `configure.in` для пакета программного обеспечения; проверяет исходные файлы в дереве каталогов, ищет в них распространенные проблемы с переносимостью и создает файл `configure.scan` , который является предварительным файлом `configure.in` для пакета.

autoupdate Изменяет файл `configure.in` , вызывающий макросы **autoconf** по их старым именам для использования текущих имен макросов

ifnames Помогает при написании файла `configure.in` для пакета; выводит идентификаторы, которые использует пакет в условных выражениях препроцессора C. (Если пакет уже был настроен для некоторой переносимости, эта программа может помочь определить,

что нужно проверить сценарию **configure**. Он также может заполнить пробелы в файле `configure.in` , сгенерированном командой **autoscan**.)

8.48. Automake-1.17

Пакет Automake содержит программы генерации Makefile для использования с Autoconf.

Приблизительное время сборки:	менее 0.1 SBU (около 1.6 SBU с тестами)
Требуемое дисковое пространство:	121 MB

8.48.1. Установка пакета Automake

Подготовьте Automake к компиляции:

```
./configure --prefix=/usr --docdir=/usr/share/doc/automake-1.17
```

Скомпилируйте пакет:

```
make
```

Использование четырех параллельных заданий ускоряет тестирование даже на системах с меньшим количеством логических ядер из-за внутренних задержек в отдельных тестах. Чтобы протестировать пакет, выполните:

```
make -j$((($(nproc)>4?$(nproc):4)) check
```

Замените *$((...))* количеством логических ядер, которые вы хотите использовать, если вы не планируете использовать все.

Установите пакет:

```
make install
```

8.48.2. Содержимое пакета Automake

Установленные программы:	aclocal, aclocal-1.17 (жестко связан с aclocal), automake, и automake-1.17 (жестко связан с automake)
Созданные каталоги:	/usr/share/aclocal-1.17, /usr/share/automake-1.17, и /usr/share/doc/automake-1.17

Краткое описание

aclocal	Генерирует файлы `aclocal.m4` на основе содержимого файла `configure.in`
aclocal-1.17	Жесткая ссылка на **aclocal**
automake	Инструмент для автоматического создания `Makefile.in` из файлов `Makefile.am` [Чтобы создать все файлы `Makefile.in` запустите эту программу в каталоге верхнего уровня. Сканируя файл `configure.in` , он автоматически находит все подходящие файлы `Makefile.am` и создает соответствующий `Makefile.in` .]
automake-1.17	Жесткая ссылка на **automake**

8.49. OpenSSL-3.3.1

Пакет OpenSSL содержит инструменты управления и библиотеки, относящиеся к криптографии. Они полезны для предоставления криптографических функций другим пакетам, таким как OpenSSH, приложениям электронной почты и веб-браузерам (для доступа к сайтам по HTTPS).

Приблизительное время сборки: 1.7 SBU
Требуемое дисковое пространство: 883 MB

8.49.1. Установка пакета OpenSSL

Подготовьте OpenSSL к компиляции:

```
./config --prefix=/usr         \
         --openssldir=/etc/ssl \
         --libdir=lib          \
         shared                \
         zlib-dynamic
```

Скомпилируйте пакет:

```
make
```

Чтобы протестировать пакет, выполните:

```
HARNESS_JOBS=$(nproc) make test
```

Известно, что один тест, 30-test_afalg.t, завершится ошибкой, если в ядре хоста не включен параметр `CONFIG_CRYPTO_USER_API_SKCIPHER` или отсутствуют какие-либо опции, обеспечивающих реализацию AES с CBC (например, комбинация `CONFIG_CRYPTO_AES` и `CONFIG_CRYPTO_CBC` или `CONFIG_CRYPTO_AES_NI_INTEL`, если процессор поддерживает AES-NI). В случае неудачи его можно смело игнорировать.

Установите пакет:

```
sed -i '/INSTALL_LIBS/s/libcrypto.a libssl.a//' Makefile
make MANSUFFIX=ssl install
```

Добавьте версию к имени каталога документации, чтобы структура соответствовала другим пакетам:

```
mv -v /usr/share/doc/openssl /usr/share/doc/openssl-3.3.1
```

По желанию, установите дополнительную документацию:

```
cp -vfr doc/* /usr/share/doc/openssl-3.3.1
```

Примечание

Вы должны обновить OpenSSL, когда будет выпущена новая версия, исправляющая уязвимости. Начиная с OpenSSL 3.0.0, схема управления версиями OpenSSL следует формату MAJOR.MINOR.PATCH. Совместимость API/ABI гарантируется для одной и той же ОСНОВНОЙ (MAJOR) версии. Поскольку LFS устанавливает только общие библиотеки, нет необходимости перекомпилировать пакеты, которые ссылаются на `libcrypto.so` или `libssl.so`, *при обновлении до версии с тем же ОСНОВНЫМ номером версии*.

Все запущенные программы, связанные с этими библиотеками, после обновления необходимо остановить и перезапустить. Для получения более подробной информации ознакомьтесь с соответствующей записью в Раздел 8.2.1, «Проблемы с обновлением».

8.49.2. Установка OpenSSL - 32-битная версия

Выполните очистку после предыдущей сборки:

```
make distclean
```

Подготовьте OpenSSL к компиляции:

```
./config --prefix=/usr         \
         --openssldir=/etc/ssl \
         --libdir=lib32        \
         shared                \
         zlib-dynamic          \
         linux-x86
```

Скомпилируйте пакет:

```
make
```

Установите пакет:

```
make DESTDIR=$PWD/DESTDIR install
cp -Rv DESTDIR/usr/lib32/* /usr/lib32
rm -rf DESTDIR
```

8.49.3. Содержимое пакета OpenSSL

Установленные программы:	c_rehash и openssl
Установленные библиотеки:	libcrypto.so и libssl.so
Созданные каталоги:	/etc/ssl, /usr/include/openssl, /usr/lib/engines и /usr/share/doc/openssl-3.3.1

Краткое описание

c_rehash	это Perl скрипт, который сканирует все файлы в каталоге и добавляет символические ссылки к их хеш-значениям. Использование **c_rehash** считается устаревшим и должно быть заменено командой **openssl rehash**
openssl	это инструмент командной строки для использования различных криптографических функций библиотеки OpenSSL из оболочки. Его можно использовать для различных функций, которые задокументированы в *openssl(1)*
`libcrypto.so`	реализует широкий спектр криптографических алгоритмов, используемых в различных интернет-стандартах. Услуги, предоставляемые этой библиотекой, используют OpenSSL-реализацию SSL, TLS и S/MIME, а также для реализации OpenSSH, OpenPGP и других криптографических стандартов.
`libssl.so`	реализует протокол безопасности транспортного уровня (TLS v1). Он предоставляет богатый API, документацию по которому можно найти в руководстве *ssl(7)*

8.50. Kmod-33

Пакет Kmod содержит библиотеки и утилиты для загрузки модулей ядра.

Приблизительное время сборки: менее 0.1 SBU

Требуемое дисковое пространство: 11 MB

8.50.1. Установка пакета Kmod

Подготовьте Kmod к компиляции:

```
./configure --prefix=/usr     \
            --sysconfdir=/etc \
            --with-openssl    \
            --with-xz         \
            --with-zstd       \
            --with-zlib       \
            --disable-manpages
```

Значение параметров настройки:

`--with-openssl`
Этот параметр позволяет Kmod обрабатывать сигнатуры PKCS7 для модулей ядра.

`--with-xz`, *`--with-zlib`* и *`--with-zstd`*
Эти параметры позволяют Kmod обрабатывать сжатые модули ядра.

`--disable-manpages`
Эта опция отключает генерацию справочных страниц, для которых требуется внешняя программа.

Скомпилируйте пакет:

```
make
```

Набору тестов этого пакета необходимы необработанные заголовочные файлы ядра (а не «очищенные», установленные ранее), но это выходит за рамки LFS.

Установите пакет и восстановите некоторые символические ссылки для совместимости с Module-Init-Tools (пакетом, который ранее обрабатывал модули ядра Linux). Система сборки создаст все эти символические ссылки в `/usr/bin` , но здесь нам нужна только команда **lsmod**, а все остальные символические ссылки должны вести в `/usr/sbin` :

```
make install

for target in depmod insmod modinfo modprobe rmmod; do
  ln -sfv ../bin/kmod /usr/sbin/$target
  rm -fv /usr/bin/$target
done
```

8.50.2. Установка Kmod - 32-битная версия

Выполните очистку после предыдущей сборки, но сохраните справочные страницы, поскольку их невозможно воссоздать, так как xsltproc еще не установлен:

```
sed -e "s/^CLEANFILES =.*/CLEANFILES =/" -i man/Makefile
make clean
```

Подготовьте Kmod к компиляции:

```
CC="gcc -m32" ./configure \
    --host=i686-pc-linux-gnu      \
    --prefix=/usr                 \
    --libdir=/usr/lib32           \
    --sysconfdir=/etc             \
    --with-openssl                \
    --with-xz                     \
    --with-zstd                   \
    --with-zlib                   \
    --disable-manpages            \
    --with-rootlibdir=/usr/lib32
```

Скомпилируйте пакет:

```
make
```

Установите пакет:

```
make DESTDIR=$PWD/DESTDIR install
cp -Rv DESTDIR/usr/lib32/* /usr/lib32
rm -rf DESTDIR
```

8.50.3. Содержимое пакета Kmod

Установленные программы:	depmod (ссылка на kmod), insmod (ссылка на kmod), kmod, lsmod (ссылка на kmod), modinfo (ссылка на kmod), modprobe (ссылка на kmod) и rmmod (ссылка на kmod)
Установленные библиотеки:	libkmod.so

Краткое описание

depmod	Создает файл зависимостей на основе символов найденных в существующем наборе модулей; этот файл используется программой **modprobe** для автоматической загрузки необходимых модулей
insmod	Устанавливает загружаемый модуль в работающее ядро
kmod	Загружает и выгружает модули ядра
lsmod	Список загруженных в данный момент модулей
modinfo	Проверяет объектный файл, связанный с модулем ядра, и отображает всю информацию, которую он смог собрать.
modprobe	Использует файл зависимостей, созданный **depmod**, для автоматической загрузки соответствующих модулей
rmmod	Выгружает модули из работающего ядра
`libkmod`	Библиотека используемая другими программами для загрузки и выгрузки модулей ядра

8.51. Libelf из Elfutils-0.191

Libelf — это библиотека для обработки файлов ELF (Executable and Linkable Format - формат исполняемых и связываемых файлов).

Приблизительное время сборки: 0.3 SBU
Требуемое дисковое пространство: 127 MB

8.51.1. Установка пакета Libelf

Libelf является частью пакета elfutils-0.191. Используйте elfutils-0.191.tar.bz2 в качестве исходного архива.

Подготовьте Libelf к компиляции:

```
./configure --prefix=/usr                \
            --disable-debuginfod         \
            --enable-libdebuginfod=dummy
```

Скомпилируйте пакет:

```
make
```

Чтобы протестировать пакет, выполните:

```
make check
```

Установите только Libelf:

```
make -C libelf install
install -vm644 config/libelf.pc /usr/lib/pkgconfig
rm /usr/lib/libelf.a
```

8.51.2. Установка Libelf - 32-битная версия

Выполните очистку после предыдущей сборки:

```
make distclean
```

Подготовьте Libelf к компиляции:

```
CC="gcc -m32" CXX="g++ -m32" ./configure \
    --host=i686-pc-linux-gnu \
    --prefix=/usr            \
    --libdir=/usr/lib32      \
    --disable-debuginfod     \
    --enable-libdebuginfod=dummy
```

Скомпилируйте пакет:

```
make
```

Установите пакет:

```
make DESTDIR=$PWD/DESTDIR -C libelf install
install -vDm644 config/libelf.pc DESTDIR/usr/lib32/pkgconfig/libelf.pc
cp -Rv DESTDIR/usr/lib32/* /usr/lib32
rm -rf DESTDIR
```

8.51.3. Содержимое пакета Libelf

Установленные библиотеки: libelf.so
Созданные каталоги: /usr/include/elfutils

Краткое описание

`libelf.so`	Содержит функции API для обработки объектных файлов ELF

8.52. Libffi-3.4.6

Библиотека Libffi предоставляет переносимый высокоуровневый программный интерфейс для различных соглашений о вызовах. Это позволяет программисту вызывать любую функцию, указанную в описании интерфейса вызова во время выполнения.

FFI расшифровывается как интерфейс внешних функций. FFI позволяет программе, написанной на одном языке, вызывать программу, написанную на другом языке. В частности, Libffi может обеспечить связь между интерпретатором, таким как Perl или Python, и подпрограммами общей библиотеки, написанными на C или C++.

Приблизительное время сборки: 1.7 SBU
Требуемое дисковое пространство: 11 MB

8.52.1. Установка пакета Libffi

Примечание

Как и GMP, Libffi собирается с учетом оптимизаций, специфичных для используемого процессора. При сборке для другой системы измените значение параметра *--with-gcc-arch=* в следующей команде на имя архитектуры, полностью реализованной процессором в этой системе. Если этого не сделать, все приложения, ссылающиеся на `libffi`, будут вызывать ошибку «Illegal Operation - недопустимая операция».

Подготовьте Libffi к компиляции:

```
./configure --prefix=/usr          \
            --disable-static       \
            --with-gcc-arch=native
```

Значение параметров configure:

--with-gcc-arch=native
Убедитесь, что GCC оптимизируется для текущей системы. Если значение не указано, то архитектура системы угадывается и сгенерированный код может быть неправильным. Если сгенерированный код будет скопирован из родной системы в менее мощную, используйте архитектуру менее мощной системы в качестве параметра. Дополнительные сведения об альтернативных типах систем смотрите в *описании параметров x86 в руководстве GCC*.

Скомпилируйте пакет:

```
make
```

Чтобы протестировать пакет, выполните:

```
make check
```

Установите пакет:

```
make install
```

8.52.2. Установка Libffi - 32-битная версия

Выполните очистку после предыдущей сборки:

```
make distclean
```

Подготовьте Libffi к компиляции:

```
CC="gcc -m32" CXX="g++ -m32" ./configure \
    --host=i686-pc-linux-gnu \
    --prefix=/usr            \
    --libdir=/usr/lib32      \
    --disable-static         \
    --with-gcc-arch=i686
```

Скомпилируйте пакет:

```
make
```

Чтобы протестировать пакет, выполните:

```
make check
```

Установите пакет:

```
make DESTDIR=$PWD/DESTDIR install
cp -Rv DESTDIR/usr/lib32/* /usr/lib32
rm -rf DESTDIR
```

8.52.3. Содержимое пакета Libffi

Установленные библиотеки: libffi.so

Краткое описание

`libffi` Содержит внешний интерфейс для API-функций

8.53. Python-3.12.5

Пакет Python 3 содержит среду разработчика Python. Его можно использовать для объектно-ориентированного программирования, написания скриптов, прототипирования больших программ и разработка целых приложений. Python — это интерпретируемый язык программирования.

Приблизительное время сборки: 2.2 SBU
Требуемое дисковое пространство: 530 MB

8.53.1. Установка пакета Python 3

Подготовьте Python к компиляции:

```
./configure --prefix=/usr        \
            --enable-shared      \
            --with-system-expat  \
            --enable-optimizations
```

Значение параметров настройки:

`--with-system-expat`
Этот параметр выполняет линковку с системной версией Expat.

`--enable-optimizations`
Этот параметр позволяет выполнить обширные, но отнимающие много времени, действия по оптимизации. Интерпретатор собирается дважды; тесты, выполненные при первой сборке, используются для улучшения финальной версии.

Скомпилируйте пакет:

```
make
```

Известно, что некоторые тесты **Python** иногда зависают на неопределенный срок. Чтобы протестировать пакет, запустите набор тестов, установив 2-минутный лимит времени для каждого тестового примера:

```
make test TESTOPTS="--timeout 120"
```

На медленной системе вам может потребоваться ещё увеличить ограничение по времени, и 1 SBU (измеряется при сборке Binutils - Проход 1 с использованием одного ядра ЦП) должно быть достаточно. В процессе выполнения некоторых тестов бывают сбои, поэтому набор тестов автоматически перезапускает неудачно завершённые тесты. Если сначала тест не пройден, но пройден при повторном запуске, его следует считать пройденным.

Установите пакет:

```
make install
```

В некоторых местах книги, мы используем команду **pip3** для установки программ и модулей Python 3 от имени пользователя `root`. Это противоречит рекомендации разработчиков Python: устанавливать пакеты в виртуальную среду или домашний каталог обычного пользователя (путем запуска **pip3** от имени этого пользователя). Поэтому всякий раз при использовании **pip3** от имени пользователя `root` появляется многострочное предупреждение.

Основная причина этой рекомендации — избежать конфликта с системным менеджером пакетов (например, **dpkg**), но в LFS нет общесистемного менеджера пакетов, так что это не проблема. Кроме того, **pip3** будет пытаться проверять наличие новой версии при каждом запуске. Поскольку разрешение доменных имен в среде chroot LFS еще не настроено, он не сможет проверить наличие новой версии и выдаст предупреждение.

Как только мы загрузим систему LFS и настроим сетевое подключение, **pip3** выдаст предупреждение, сообщающее пользователю о необходимости обновить его с помощью предварительно собранного whl-файла в PyPI (всякий раз, когда будет доступна новая версия). Но LFS считает **pip3** частью Python3, поэтому его не следует обновлять отдельно. Кроме того, обновление из whl-файла не соответствует цели проекта — собрать систему Linux из исходного кода, поэтому предупреждение о новой версии **pip3** следует игнорировать. По желанию, вы можете отключить все предупреждение, создав следующий файл конфигурации:

```
cat > /etc/pip.conf << EOF
[global]
root-user-action = ignore
disable-pip-version-check = true
EOF
```

Важно

В LFS и BLFS мы собираем и устанавливаем модули Python с помощью команды **pip3**. Убедитесь, что команда **pip3 install** в обеих книгах запускаются от имени пользователя `root` (если только она не для виртуальной среды Python). Запуск **pip3 install** от имени пользователя без полномочий `root` может показаться нормальным, но это приведет к тому, что установленный модуль будет недоступен для других пользователей.

Команда **pip3 install** по умолчанию не приведёт к автоматической переустановке уже установленного модуля. Чтобы использовать команду **pip3 install** для обновления модуля (например, с meson-0.61.3 до meson-0.62.0), добавьте параметр *`--upgrade`* в командную строку. Если по какой-то причине необходимо понизить версию модуля или переустановить ту же версию, используйте параметр *`--force-reinstall --no-deps`* .

По желанию установите предварительно отформатированную документацию:

```
install -v -dm755 /usr/share/doc/python-3.12.5/html

tar --no-same-owner \
    -xvf ../python-3.12.5-docs-html.tar.bz2
cp -R --no-preserve=mode python-3.12.5-docs-html/* \
    /usr/share/doc/python-3.12.5/html
```

Значение команд установки документации:

`--no-same-owner` (tar) и `--no-preserve=mode` (cp)

Параметры обеспечивают, что установленные файлы имеют корректные права и владельца файлов. Использование tar без этих параметров приведет к установке файлов с правами пользователя создавшего архив и файлы будут иметь ограниченные разрешения.

8.53.2. Содержимое пакета Python 3

Установленные программы:	2to3, idle3, pip3, pydoc3, python3 и python3-config
Установленные библиотеки:	libpython3.12.so и libpython3.so
Созданные каталоги:	/usr/include/python3.12, /usr/lib/python3 и /usr/share/doc/python-3.12.5

Краткое описание

2to3	программа на Python, которая читает файлы написанные на Python 2.x, применяет к ним серию изменений и переводит их в валидный код Python 3.x.

idle3 скрипт-обертка, который открывает графический редактор с поддержкой Python. Для запуска этого скрипта, перед установкой Python необходимо установить Tk, чтобы модуль Tkinter Python был собран.

pip3 Установщик пакетов для Python. Вы можете использовать pip для установки пакетов из каталога PyPI (Python Package Index) и других источников.

pydoc3 инструмент документации Python

python3 это интерпретатор для Python, интерпретируемый, интерактивный, объектно-ориентированный язык программирования

8.54. Flit-Core-3.9.0

Flit-core — это часть Flit, предназначенная для сборки дистрибутива (инструмента для упаковки простых модулей Python).

Приблизительное время сборки: менее 0.1 SBU

Требуемое дисковое пространство: 1.6 MB

8.54.1. Установка пакета Flit-Core

Соберите пакет:

```
pip3 wheel -w dist --no-cache-dir --no-build-isolation --no-deps $PWD
```

Установите пакет:

```
pip3 install --no-index --no-user --find-links dist flit_core
```

Значение параметров конфигурации и команд pip3:

wheel
Эта команда создает архив wheel для этого пакета.

-w dist
Указывает pip поместить созданный архив в каталог `dist`.

--no-cache-dir
Не позволяет pip копировать созданный wheel-пакет в каталог `/root/.cache/pip` .

install
Эта команда устанавливает пакет.

--no-build-isolation, *--no-deps* и *--no-index*
Эти параметры предотвращают получение файлов из онлайн-репозитория пакетов (PyPI). Если пакеты установлены в правильном порядке, то нет необходимости загружать какие-либо файлы; эти параметры усиливают безопасность в случае ошибки пользователя.

--find-links dist
Указывает pip искать архивы wheel в каталоге `dist`.

8.54.2. Содержимое пакета Flit-Core

Созданные каталоги: /usr/lib/python3.12/site-packages/flit_core и /usr/lib/python3.12/site-packages/flit_core-3.9.0.dist-info

8.55. Wheel-0.44.0

Wheel — это библиотека Python, которая является эталонной реализацией стандарта упаковки программ на языке Python.

Приблизительное время сборки: менее 0.1 SBU
Требуемое дисковое пространство: 1.5 MB

8.55.1. Установка пакета Wheel

Скомпилируйте Wheel с помощью следующей команды:

```
pip3 wheel -w dist --no-cache-dir --no-build-isolation --no-deps $PWD
```

Установите Wheel:

```
pip3 install --no-index --find-links=dist wheel
```

8.55.2. Содержимое пакета Wheel

Установленные программы: wheel
Созданные каталоги: /usr/lib/python3.12/site-packages/wheel и /usr/lib/python3.12/site-packages/wheel-0.44.0.dist-info

Краткое описание

wheel это утилита для распаковки, упаковки или преобразования wheel-архивов

8.56. Setuptools-72.2.0

Пакет Setuptools это инструмент, используемый для загрузки, сборки, установки, обновления и удаления пакетов Python.

Приблизительное время сборки: менее 0.1 SBU
Требуемое дисковое пространство: 35 MB

8.56.1. Установка пакета Setuptools

Соберите пакет:

```
pip3 wheel -w dist --no-cache-dir --no-build-isolation --no-deps $PWD
```

Установите пакет:

```
pip3 install --no-index --find-links dist setuptools
```

8.56.2. Содержимое пакета Setuptools

Созданные каталоги: /usr/lib/python3.12/site-packages/_distutils_hack, /usr/lib/python3.12/site-packages/pkg_resources, /usr/lib/python3.12/site-packages/setuptools, and /usr/lib/python3.12/site-packages/setuptools-72.2.0.dist-info

8.57. Ninja-1.12.1

Ninja - небольшая система сборки ориентированная на скорость.

Приблизительное время сборки: 0.2 SBU
Требуемое дисковое пространство: 37 MB

8.57.1. Установка пакета Ninja

При запуске **ninja** обычно использует максимальное количество процессов параллельно. По умолчанию это количество ядер в системе плюс два. В некоторых случаях это может привести к перегреву процессора или нехватке памяти в системе. Когда **ninja** вызывается из командной строки, передача параметра -jN ограничит количество параллельных процессов. Некоторые пакеты встраивают выполнение **ninja** и параметр -j не передается.

Использование приведенной ниже *необязательной* процедуры позволяет пользователю ограничить количество параллельных процессов с помощью переменной окружения NINJAJOBS. **Пример**, настройки:

```
export NINJAJOBS=4
```

ограничит **ninja** четырьмя параллельными процессами.

По желанию, добавьте возможность использовать переменную окружения NINJAJOBS, выполнив следующую команду:

```
sed -i '/int Guess/a \
  int   j = 0;\
  char* jobs = getenv( "NINJAJOBS" );\
  if ( jobs != NULL ) j = atoi( jobs );\
  if ( j > 0 ) return j;\
' src/ninja.cc
```

Соберите Ninja с помощью команды:

```
python3 configure.py --bootstrap
```

Значение параметров сборки:

--bootstrap
Этот параметр перестраивает Ninja под текущую систему.

Тесты этого пакета не могут быть выполнены в среде chroot, так как для них требуется *cmake*.

Установите пакет:

```
install -vm755 ninja /usr/bin/
install -vDm644 misc/bash-completion /usr/share/bash-completion/completions/ninja
install -vDm644 misc/zsh-completion  /usr/share/zsh/site-functions/_ninja
```

8.57.2. Содержимое пакета Ninja

Установленные программы: ninja

Краткое описание

ninja это система сборки Ninja

8.58. Meson-1.5.1

Meson — это система сборки с открытым исходным кодом, разработанная таким образом, чтобы быть очень быстрой и максимально удобной для пользователя.

Приблизительное время сборки: менее 0.1 SBU
Требуемое дисковое пространство: 43 MB

8.58.1. Установка пакета Meson

Скомпилируйте Meson с помощью следующей команды:

```
pip3 wheel -w dist --no-cache-dir --no-build-isolation --no-deps $PWD
```

Для набора тестов требуются некоторые пакеты, выходящие за рамки LFS.

Установите пакет:

```
pip3 install --no-index --find-links dist meson
install -vDm644 data/shell-completions/bash/meson /usr/share/bash-completion/completions/meson
install -vDm644 data/shell-completions/zsh/_meson /usr/share/zsh/site-functions/_meson
```

Значение параметров установки:

`-w dist`
Помещает собранный wheels в каталог `dist`.

`--find-links dist`
Устанавливает wheels из каталога `dist`.

8.58.2. Содержимое пакета Meson

Установленные программы: meson
Созданные каталоги: /usr/lib/python3.12/site-packages/meson-1.5.1.dist-info и /usr/lib/python3.12/site-packages/mesonbuild

Краткое описание

meson Высокопроизводительная система сборки

8.59. Coreutils-9.5

Пакет Coreutils содержит основные утилиты, необходимые каждой операционной системе.

Приблизительное время сборки:	1.1 SBU
Требуемое дисковое пространство:	173 MB

8.59.1. Установка пакета Coreutils

Стандарт POSIX требует, чтобы программы пакета Coreutils правильно распознавали символы даже в случае, если используются многобайтовые локали. Следующий патч исправляет несоответствие этому требованию, а также другие ошибки, касающиеся интернационализации:

```
patch -Np1 -i ../coreutils-9.5-i18n-2.patch
```

Примечание

В этом патче было обнаружено много ошибок. Сообщая о новых ошибках разработчикам Coreutils, сначала проверьте, воспроизводятся ли эти ошибки без этого исправления.

Теперь подготовьте Coreutils к компиляции:

```
autoreconf -fiv
FORCE_UNSAFE_CONFIGURE=1 ./configure \
            --prefix=/usr            \
            --enable-no-install-program=kill,uptime
```

Значение параметров настройки:

autoreconf
Патч для интернационализации изменил систему сборки пакета, поэтому файлы конфигурации необходимо сгенерировать заново.

`FORCE_UNSAFE_CONFIGURE=1`
Эта переменная среды позволяет собрать пакет от имени пользователя `root`.

`--enable-no-install-program=kill,uptime`
Назначение этого параметра — запретить Coreutils устанавливать программы, которые будут установлены другими пакетами.

Скомпилируйте пакет:

```
make
```

Если вы не планируете запускать набор тестов, перейдите к разделу «Установка пакета».

Теперь набор тестов готов к запуску. Сначала запустите тесты, предназначенные для запуска от имени пользователя `root`:

```
make NON_ROOT_USERNAME=tester check-root
```

Мы собираемся выполнить остальные тесты от имени пользователя `tester`. Некоторые тесты требуют, чтобы пользователь был членом более чем одной группы. Чтобы эти тесты не были пропущены, добавьте временную группу и включите в неё пользователя `tester`:

```
groupadd -g 102 dummy -U tester
```

Исправьте некоторые разрешения, чтобы пользователь без полномочий `root` мог компилировать и запускать тесты:

```
chown -R tester .
```

Теперь запустите тесты (используя `/dev/null` для стандартного ввода, иначе два теста могут быть не пройдены при сборке LFS в графическом терминале, или сеансе SSH, или GNU Screen, поскольку стандартный ввод подключен к PTY из хост-дистрибутива, а узел устройства для такого PTY недоступен из среды chroot LFS):

```
su tester -c "PATH=$PATH make -k RUN_EXPENSIVE_TESTS=yes check" \
   < /dev/null
```

Удалить временную группу:

```
groupdel dummy
```

Известно, что два теста, `tests/cp/preserve-mode.sh` и `tests/mv/acl.sh`, завершаются неудачно в среде chroot, но проходят успешно в собранной системе.

Установите пакет:

```
make install
```

Переместите программы туда, где они должны быть в соответствие со спецификациями FHS:

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

8.59.2. Содержимое пакета Coreutils

Установленные программы:	[, b2sum, base32, base64, basename, basenc, cat, chcon, chgrp, chmod, chown, chroot, cksum, comm, cp, csplit, cut, date, dd, df, dir, dircolors, dirname, du, echo, env, expand, expr, factor, false, fmt, fold, groups, head, hostid, id, install, join, link, ln, logname, ls, md5sum, mkdir, mkfifo, mknod, mktemp, mv, nice, nl, nohup, nproc, numfmt, od, paste, pathchk, pinky, pr, printenv, printf, ptx, pwd, readlink, realpath, rm, rmdir, runcon, seq, sha1sum, sha224sum, sha256sum, sha384sum, sha512sum, shred, shuf, sleep, sort, split, stat, stdbuf, stty, sum, sync, tac, tail, tee, test, timeout, touch, tr, true, truncate, tsort, tty, uname, unexpand, uniq, unlink, users, vdir, wc, who, whoami и yes
Установленные библиотеки:	libstdbuf.so (in /usr/libexec/coreutils)
Созданные каталоги:	/usr/libexec/coreutils

Краткое описание

[	Это команда /usr/bin/[, которая является синонимом команды **test**
base32	Кодирует и декодирует данные в соответствии со спецификацией base32 (RFC 4648)
base64	Кодирует и декодирует данные в соответствии со спецификацией base64 (RFC 4648)
b2sum	Выводит или проверяет контрольные суммы BLAKE2 (512-битные)
basename	Удаляет любой путь и заданный суффикс из имени файла
basenc	Кодирует или декодирует данные с использованием различных алгоритмов
cat	Присоединяет файлы к стандартному выходному потоку
chcon	Изменяет контекст безопасности для файлов и каталогов
chgrp	Изменяет владельцев группы для файлов и директорий
chmod	Изменяет разрешения каждого файла на заданный режим; режим может быть либо символьным представлением вносимых изменений, либо восьмеричным числом, представляющим новые разрешения.

chown Изменяет принадлежность файлов и директорий пользователю и/или группе

chroot Запускает команду с указанным каталогом в качестве корневого каталога `/`

cksum Выводит контрольную сумму Cyclic Redundancy Check (CRC) и количество байтов для каждого указанного файла.

comm Сравнивает два отсортированных файла, выводя в три столбца уникальные и общие строки.

cp Копирует файлы

csplit Разбивает заданный файл на несколько новых файлов, разделяя их в соответствии с заданными шаблонами или номерами строк и выводя количество байтов для каждого нового файла.

cut Выдает участки строк, выбирая части в соответствии с заданными полями или позициями

date Отображает текущее дату и время в заданном формате или устанавливает системные дату и время

dd Копирует файл, используя заданный размер блока и количество, при необходимости выполняя преобразования на нем.

df Сообщает объем доступного (и используемого) дискового пространства во всех смонтированных файловых системах или только в файловых системах, содержащих выбранные файлы.

dir Выводит содержимое заданного каталога (так же, как команда **ls**)

dircolors Выводит команды для установки переменной среды `LS_COLOR` для изменения цветовой схемы, используемой **ls**.

dirname Извлекает часть(части) каталога из заданного(заданных) имени(имён)

du Сообщает объем дискового пространства, используемого текущим каталогом, каждым из заданных каталогов (включая все подкаталоги) или каждым из заданных файлов.

echo Отображает указанные строки

env Запускает команду в модифицированной среде окружения

expand Конвертирует символы табуляции в пробелы

expr Вычисляет выражения

factor Выводит простые множители указанных целых чисел

false Ничего не делает, указывает на неудачу; всегда завершается с кодом состояния, указывающим на сбой

fmt Форматирует абзацы в указанных файлах

fold Выполняет перенос строк в указанных файлах

groups Сообщает о принадлежности пользователя к группам

head Выводит первые десять строк (или заданное количество строк) каждого заданного файла.

hostid Выводит числовой идентификатор хоста (в шестнадцатеричном формате)

id Выводит действующий идентификатор пользователя, идентификатор группы и принадлежность к группам для текущего или для указанного пользователя

install Копирует файлы, одновременно устанавливая для них права доступа, и, если возможно, устанавливая для них владельца и группу

join Объединяет строки, которые имеют идентичные объединяемые поля в двух различных файлах

link Создает жесткую ссылку (с указанным именем) на файл

ln	Создает жесткие или мягкие (символические) ссылки между файлами
logname	Сообщает имя входа текущего пользователя
ls	Выводит список содержимого для каждого заданного каталога
md5sum	Выводит или проверяет контрольные суммы Message Digest 5 (MD5)
mkdir	Создает директории с указанными именами
mkfifo	Создает "именованный канал" "первым пришел — первым ушел" (FIFO), в нотации UNIX с заданными именами
mknod	Создает узлы устройств с заданными именами; узел устройства представляет собой специальный символьный файл, специальный файл блока или FIFO.
mktemp	Создает временные файлы безопасным способом; используется в скриптах
mv	Перемещает или переименовывает файлы или каталоги
nice	Запускает программу с измененным приоритетом исполнения
nl	Нумерует строки в указанных файлах
nohup	Запускает команду, невосприимчивую к зависаниям, а ее вывод перенаправляется в файл журнала
nproc	Выводит количество дочерних процессов, доступных для процесса.
numfmt	Преобразует числа в или из удобочитаемых строк
od	Вывод дампа файла в восьмеричном и других форматах
paste	Объединяет указанные файлы, последовательно соединяя соответствующие строки рядом друг с другом, разделенные символами табуляции.
pathchk	Проверяет, являются ли имена файлов допустимыми или переносимыми
pinky	Легковесный клиент типа finger; выдает некоторую информацию о заданных пользователях
pr	Разбивает файлы для печати на страницы и столбцы
printenv	Выдает значения переменных окружения
printf	Выводит аргументы в соответствии с заданным форматом, подобно функции C printf.
ptx	Создает перестановочный индекс по содержимому указанных файлов с каждым ключевым словом в своем контексте
pwd	Сообщает имя текущего рабочего каталога
readlink	Выдает значение указанной символической ссылки
realpath	Возвращает приведенное к обычному виду полное имя файла
rm	Удаляет файлы или каталоги
rmdir	Удаляет каталоги, если они пусты
runcon	Запускает команду с указанным контекстом безопасности
seq	Выдает последовательность чисел из указанного диапазона с указанным значением приращения
sha1sum	Выводит или проверяет контрольные суммы 160-битного алгоритма безопасного хеширования 1 (SHA1)
sha224sum	Выводит или проверяет контрольные суммы 224-битного алгоритма безопасного хеширования
sha256sum	Выводит или проверяет контрольные суммы 256-битного алгоритма безопасного хеширования

sha384sum	Выводит или проверяет контрольные суммы 384-битного алгоритма безопасного хеширования
sha512sum	Выводит или проверяет контрольные суммы 512-битного алгоритма безопасного хеширования
shred	Многократно перезаписывает заданные файлы сложными шаблонами, что затрудняет восстановление данных.
shuf	Перемешивает строки текста
sleep	Делает паузу на заданный промежуток времени
sort	Сортирует строки в указанных файлах
split	Разбивает заданный файл на несколько частей в соответствии с указанным размером или количеством строк
stat	Отображает статус файла или файловой системы
stdbuf	Запускает команды с измененными операциями буферизации для своих стандартных потоков.
stty	Устанавливает или сообщает настройки терминала
sum	Выводит контрольную сумму и количество блоков для каждого заданного файла
sync	Сбрасывает буферы файловой системы; он принудительно записывает измененные блоки на диск и обновляет суперблок
tac	Конкатенация содержимого указанных файлов в обратном порядке
tail	Выводит последние десять строк (или заданное количество строк) каждого указанного файла
tee	Считывает данные со стандартного потока ввода, записывает как в стандартный вывод, так и в указанные файлы
test	Сравнивает значения и проверяет типы файловs
timeout	Запускает команду с ограничением по времени
touch	Изменяет временные метки файлов, устанавливая время доступа и модификации данных файлов на текущее время; несуществующие файлы создаются с нулевой длиной
tr	Переводит, сжимает и удаляет заданные символы из стандартного потока
true	Ничего не делает, указывает на успешное выполнение операции; он всегда завершается с кодом состояния, указывающим на успех
truncate	Сжимает или расширяет файл до указанного размера
tsort	Выполняет топологическую сортировку; записывает полностью упорядоченный список в соответствии с частичным упорядочением в данном файле
tty	Сообщает имя файла терминала, подключенного к стандартному вводу.
uname	Сообщает системную информацию
unexpand	Преобразует пробелы в табуляции
uniq	Удаляет все повторяющиеся копии уже имеющихся строк, кроме одной
unlink	Удаляет указанный файл
users	Сообщает имена пользователей, вошедших в систему в данный момент
vdir	То же, что **ls -l**
wc	Сообщает количество строк, слов и байт для каждого заданного файла, а также общее количество строк, если указано более одного файла
who	Сообщает, кто вошел в систему

whoami Сообщает имя пользователя, соответствующее идентификатору текущего пользователя

yes Повторно выводит `y` или указанную строку, до тех пор, пока команда не будет завершена с помощью kill

`libstdbuf` Библиотека, используемая командой **stdbuf**

8.60. Check-0.15.2

Check - это фреймворк модульного тестирования для языка C.

Приблизительное время сборки:	0.1 SBU (около 2.1 SBU с тестами)
Требуемое дисковое пространство:	11 MB

8.60.1. Установка пакета Check

Подготовить Check к компиляции:

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

Соберите пакет:

```
make
```

Когда компиляция будет завершена, запустите набор тестов:

```
make check
```

Установите пакет:

```
make docdir=/usr/share/doc/check-0.15.2 install
```

8.60.2. Содержимое пакета Check

Установленные программы:	checkmk
Установленные библиотеки:	libcheck.so

Краткое описание

checkmk	Сценарий Awk для генерации unit-тестов C, для использования с платформой модульного тестирования Check.
`libcheck.so`	Содержит функции, позволяющие вызывать Check из программы тестирования.

8.61. Diffutils-3.10

Пакет Diffutils содержит программы, которые показывают различия между файлами или каталогами.

Приблизительное время сборки:	0.3 SBU
Требуемое дисковое пространство:	42 MB

8.61.1. Установка пакета Diffutils

Подготовьте Diffutils к компиляции:

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

Скомпилируйте пакет:

```
make
```

Чтобы протестровать пакет, выполните:

```
make check
```

Установите пакет:

```
make install
```

8.61.2. Содержимое пакета Diffutils

Установленные программы:	cmp, diff, diff3, и sdiff

Краткое описание

cmp	Сравнивает побайтно два файла и сообщает о любых различиях
diff	Сравнивает два файла или каталога и сообщает, какие строки отличаются
diff3	Сравнивает три файла построчно
sdiff	Объединяет два файла и интерактивно выводит результат

8.62. Gawk-5.3.0

Пакет Gawk содержит программы для работы с текстовыми файлами.

Приблизительное время сборки:	0.2 SBU
Требуемое дисковое пространство:	42 MB

8.62.1. Установка пакета Gawk

Во-первых, отредактируйте Makefile, чтобы некоторые ненужные файлы не были установлены

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

Подготовьте Gawk к компиляции:

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

Скомпилируйте пакет:

```
make
```

Чтобы протестировать пакет, выполните:

```
chown -R tester .
su tester -c "PATH=$PATH make check"
```

Установите пакет:

```
rm -f /usr/bin/gawk-5.3.0
make install
```

Значение команды:

rm -f /usr/bin/gawk-5.3.0

Система сборки не будет пересоздавать жесткую ссылку `gawk-5.3.0` , если она существует. Удалите её, чтобы гарантировать, что предыдущая жесткая ссылка, созданная в Раздел 6.9, «Gawk-5.3.0», будет обновлена.

В процессе установки уже создан **awk** в виде символической ссылки на **gawk**, создайте также символическую ссылку на справочную страницу:

```
ln -sv gawk.1 /usr/share/man/man1/awk.1
```

По желанию, установите документацию:

```
mkdir -pv                                   /usr/share/doc/gawk-5.3.0
cp    -v doc/{awkforai.txt,*.{eps,pdf,jpg}} /usr/share/doc/gawk-5.3.0
```

8.62.2. Содержимое пакета Gawk

Установленные программы:	awk (ссылка на gawk), gawk и gawk-5.3.0
Установленные библиотеки:	filefuncs.so, fnmatch.so, fork.so, inplace.so, intdiv.so, ordchr.so, readdir.so, readfile.so, revoutput.so, revtwoway.so, rwarray.so и time.so (все в /usr/lib/gawk)
Созданные каталоги:	/usr/lib/gawk, /usr/libexec/awk, /usr/share/awk и /usr/share/doc/gawk-5.3.0

Краткое описание

awk	Ссылка на **gawk**

gawk	Программа для работы с текстовыми файлами; это GNU реализация **awk**
gawk-5.3.0	Жесткая ссылка на **gawk**

8.63. Findutils-4.10.0

Пакет Findutils содержит программы для поиска файлов. Эти программы предназначены для поиска по всем файлам в дереве каталогов, а также для создания, обслуживания и поиска в базе данных (часто быстрее, чем рекурсивный поиск, но ненадежно, если база данных давно не обновлялась).Findutils также предоставляет программу **xargs**, которую можно использовать для запуска указанной команды для каждого файла, выбранного при поиске.

Приблизительное время сборки:	0.7 SBU
Требуемое дисковое пространство:	62 MB

8.63.1. Установка пакета Findutils

Подготовьте Findutils к компиляции:

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

Значение параметров настройки:

`--localstatedir`
Этот параметр перемещает базу данных команды **locate** в `/var/lib/locate` , что соответствует расположению, совместимому со стандартом FHS.

Скомпилируйте пакет:

```
make
```

Чтобы протестировать пакет, выполните:

```
chown -R tester .
su tester -c "PATH=$PATH make check"
```

Установите пакет:

```
make install
```

8.63.2. Содержимое пакета Findutils

Installed programs:	Установленные программы
Созданные каталоги:	/var/lib/locate

Краткое описание

find	Выполняет поиск в заданных каталогах файлов, соответствующих критериям
locate	Выполняет поиск по базе данных имен файлов и сообщает об именах, которые содержат заданную строку или соответствуют заданному шаблону.
updatedb	Обновляет базу данных **locate**; сканирует всю файловую систему (включая другие файловые системы, которые в настоящее время смонтированы, если не указано иное) и записывает найденные имена файлов в базу данных
xargs	Может использоваться для применения заданной команды к списку файлов

8.64. Groff-1.23.0

Пакет Groff содержит программы для обработки и форматирования текста и изображений.

Приблизительное время сборки:	0.2 SBU
Требуемое дисковое пространство:	107 MB

8.64.1. Установка пакета Groff

Groff ожидает, что переменная окружения `PAGE` будет содержать размер бумаги по умолчанию. Для пользователей из США подходит *`PAGE=letter`*. Для других стран больше подойдет *`PAGE=A4`*. Хотя формат бумаги по умолчанию настраивается во время компиляции, его можно переопределить позже, записав «A4» или «letter» в файл `/etc/papersize` .

Подготовьте Groff к компиляции:

```
PAGE=<paper_size> ./configure --prefix=/usr
```

Соберите пакет:

```
make
```

Чтобы протестировать пакет, выполните:

```
make check
```

Установите пакет:

```
make install
```

8.64.2. Содержимое пакета Groff

Установленные программы:	addftinfo, afmtodit, chem, eqn, eqn2graph, gdiffmk, glilypond, gperl, gpinyin, grap2graph, grn, grodvi, groff, groffer, grog, grolbp, grolj4, gropdf, grops, grotty, hpftodit, indxbib, lkbib, lookbib, mmroff, neqn, nroff, pdfmom, pdfroff, pfbtops, pic, pic2graph, post-grohtml, preconv, pre-grohtml, refer, roff2dvi, roff2html, roff2pdf, roff2ps, roff2text, roff2x, soelim, tbl, tfmtodit и troff
Созданные каталоги:	/usr/lib/groff и /usr/share/doc/groff-1.23.0, /usr/share/groff

Краткое описание

addftinfo	Читает файл шрифта troff и добавляет некоторую дополнительную информацию о метрике шрифта, используемую системой **groff**.
afmtodit	Создаёт файл шрифта для использования с **groff** и **grops**
chem	Препроцессор Groff для создания диаграмм химических структур
eqn	Компилирует описания уравнений, имеющихся внутри входных файлов troff, которые понятны **troff**
eqn2graph	Преобразует a troff EQN (уравнение) во фрагмент изображения
gdiffmk	Отображает различия между файлами groff/nroff/troff
glilypond	Преобразует ноты, записанные на языке lilypond в язык groff
gperl	Препроцессор для groff, позволяющий вставлять код perl в файлы groff
gpinyin	Препроцессор groff, позволяющий вставлять Pinyin (запись звуков китайского языка с помощью латинского алфавита) в файлы groff.

grap2graph	Преобразует диаграммы grap во фрагмент растрового изображения (grap - это старый язык программирования Unix для создания диаграмм)
grn	Препроцессор **groff** для файлов gremlin
grodvi	Драйвер для **groff**, создающий выходные файлы в формате TeX dvi
groff	Внешний интерфейс к системе форматирования документов groff; обычно он запускает программу **troff** и постпроцессор, соответствующий выбранному устройству
groffer	Отображает файлы groff и справочные страницы на терминалах X и tty
grog	Читает файлы и пытается определить, какие из параметров **groff** `-e` , `-man` , `-me` , `-mm` , `-ms` , `-p` , `-s` , или `-t` требуются для печати файлов, и указывает команду **groff** с этими параметрами
grolbp	Драйвер **groff** для принтеров Canon CAPSL (лазерные принтеры серий LBP-4 и LBP-8)
grolj4	Драйвер для **groff** который выводит результат в формате PCL5, подходящем для принтера HP LaserJet 4
gropdf	Переводит выходные данные GNU **troff** в формат PDF
grops	Переводит выходные данные GNU **troff** в формат PostScript
grotty	Переводит вывод GNU **troff** в форму, подходящую для устройств, подобных пишущим машинкам.
hpftodit	Создает файл шрифта для использования с **groff -Tlj4** из файла метрик шрифта для HP
indxbib	Создает инвертированный индекс для библиографических баз данных для указанного файла, используемый с **refer**, **lookbib**, и **lkbib**
lkbib	Выполняет поиск в библиографических базах данных ссылок, содержащих указанные ключи, и сообщает о любых найденных ссылках
lookbib	Выводит приглашение при наличии стандартной ошибки (если устройство стандартного ввода не является терминалом), читает из устройства стандартного ввода строку, в которой находится набор ключевых слов, ищет в библиографической базе данных для указанного файла ссылки, содержащие эти ключевые слова, выводит все ссылки, найденные в стандартном выводе и повторяет этот процесс до тех пор, пока не завершится входной поток
mmroff	Простой препроцессор для **groff**
neqn	Форматирует уравнения для их вывода в формате American Standard Code for Information Interchange (ASCII)
nroff	Скрипт, который эмулирует команду **nroff** с помощью **groff**
pdfmom	Это обертка над groff которая упрощает создание PDF-документов из файлов, отформатированных с помощью макросов mom.
pdfroff	Создает pdf-документы с помощью groff
pfbtops	Преобразует шрифт PostScript в формате `.pfb` в формат ASCII
pic	Компилирует описания изображений, вставленных во входные файлы troff или TeX, в команды, понятные TeX или **troff**
pic2graph	Преобразует диаграмму PIC во фрагмент изображения
post-grohtml	Переводит выходной поток GNU **troff** в HTML
preconv	Преобразует кодировку входных файлов в формат, понимаемый GNU **troff**
pre-grohtml	Переводит выходной поток GNU **troff** в HTML

refer	Копирует содержимое файла в стандартный вывод, кроме тех символов, которые расположены между *.[* и *.]* и интерпретируются как цитаты, и кроме строк между *.R1* и *.R2*, которые интерпретируются как команды, указывающие как цитаты должны быть обработаны
roff2dvi	Преобразует файлы roff в формат DVI
roff2html	Преобразует файлы roff в формат HTML
roff2pdf	Преобразует файлы roff в формат PDF
roff2ps	Преобразует файлы roff в файлы ps
roff2text	Преобразует файлы roff в текстовые файлы
roff2x	Преобразует файлы roff в другие форматы
soelim	Читает файлы и заменяет строки вида *.so file* содержимым указанного файла *file*
tbl	Компилирует описания таблиц, вставленные во входные файлы troff, в команды, понимаемые **troff**
tfmtodit	Создает файл шрифта для использования с **groff -Tdvi**
troff	Полностью совместим с Unix **troff**; его следует вызывать с помощью команды **groff**, которая также будет запускать препроцессоры и постпроцессоры в соответствующем порядке и с соответствующими параметрами

8.65. GRUB-2.12

Пакет GRUB содержит загрузчик операционной системы от проекта GNU (GRand Unified Bootloader).

Приблизительное время сборки: 0.3 SBU
Требуемое дисковое пространство: 166 MB

8.65.1. Установка пакета GRUB

Примечание

Если ваша система поддерживает UEFI и вы хотите загрузить LFS с UEFI, вам необходимо установить GRUB с поддержкой UEFI (и его зависимости), следуя инструкциям на *страница BLFS*. Вы можете пропустить установку этого пакета или установить его и пакет из BLFS (на странице BLFS приведены инструкции для обоих случаев).

Предупреждение

Сбросьте переменные окружения, которые могут повлиять на сборку:

```
unset {C,CPP,CXX,LD}FLAGS
```

Не пытайтесь «настраивать» этот пакет с помощью пользовательских флагов компиляции. Этот пакет является загрузчиком. Низкоуровневые операции в исходном коде могут быть нарушены из-за агрессивной оптимизации.

Добавьте файл, отсутствующий в архиве релиза:

```
echo depends bli part_gpt > grub-core/extra_deps.lst
```

Подготовьте GRUB к компиляции:

```
./configure --prefix=/usr          \
            --sysconfdir=/etc      \
            --disable-efiemu       \
            --disable-werror
```

Значение новых параметров настройки:

`--disable-werror`
Этот параметр позволяет завершить сборку с предупреждениями, появившимися в более поздних версиях Flex.

`--disable-efiemu`
Этот параметр запрещает установку компонента, отключает функции и некоторые программы тестирования, которые не нужны для LFS.

Скомпилируйте пакет:

```
make
```

Запуск набора тестов не рекомендуется. Большинство тестов зависят от пакетов, недоступных в ограниченной среде LFS. Если вы все равно хотите запустить тесты, выполните **make check**.

Установите пакет:

```
make install
mv -v /etc/bash_completion.d/grub /usr/share/bash-completion/completions
```

Создание загружаемой системы LFS с помощью GRUB будет обсуждаться в Раздел 10.4, «Использование GRUB для настройки процесса загрузки».

8.65.2. Содержимое пакета GRUB

Установленные программы:	grub-bios-setup, grub-editenv, grub-file, grub-fstest, grub-glue-efi, grub-install, grub-kbdcomp, grub-macbless, grub-menulst2cfg, grub-mkconfig, grub-mkimage, grub-mklayout, grub-mknetdir, grub-mkpasswd-pbkdf2, grub-mkrelpath, grub-mkrescue, grub-mkstandalone, grub-ofpathname, grub-probe, grub-reboot, grub-render-label, grub-script-check, grub-set-default, grub-sparc64-setup и grub-syslinux2cfg
Созданные каталоги:	/usr/lib/grub, /etc/grub.d, /usr/share/grub и /boot/grub (при первом запуске grub-install)

Краткое описание

grub-bios-setup	Вспомогательная программа для **grub-install**
grub-editenv	Инструмент для редактирования блока окружения (environment block)
grub-file	Проверяет, относится ли данный файл к указанному типу
grub-fstest	Инструмент для отладки драйвера файловой системы
grub-glue-efi	Объединяет 32-разрядные и 64-разрядные бинарные файлы в один файл (для компьютеров Apple)
grub-install	Устанавливает GRUB на ваш диск
grub-kbdcomp	Скрипт, который преобразует макет xkb в макет, распознаваемый GRUB
grub-macbless	Это аналог bless в стиле Mac для файловых систем HFS или HFS+ (команда **bless** характерна для компьютеров Apple; она делает устройство загрузочным)
grub-menulst2cfg	Преобразует GRUB Legacy `menu.lst` в `grub.cfg` для использования с GRUB 2
grub-mkconfig	Генерирует файл `grub.cfg`
grub-mkimage	Создаёт загрузочный образ GRUB
grub-mklayout	Создаёт файл раскладки клавиатуры GRUB
grub-mknetdir	Подготавливает сетевой загрузочный каталог GRUB
grub-mkpasswd-pbkdf2	Генерирует зашифрованный пароль PBKDF2 для использования в меню загрузки
grub-mkrelpath	Создает имена системных путей относительно корня
grub-mkrescue	Создает загрузочный образ GRUB, подходящий для дискеты, CDROM/DVD или USB-накопителя
grub-mkstandalone	Генерирует автономный образ
grub-ofpathname	Вспомогательная программа, которая выводит путь к устройству GRUB
grub-probe	Проверяет информацию об устройстве для заданного пути или устройства
grub-reboot	Устанавливает пункт меню в GRUB для загрузки по умолчанию, только для следующей загрузки(однократно)
grub-render-label	Отображает .disk_label для компьютеров Apple Mac
grub-script-check	Проверяет скрипт настройки GRUB на наличие синтаксических ошибок
grub-set-default	Устанавливает для GRUB загрузочную запись по умолчанию
grub-sparc64-setup	Вспомогательная программа для grub-setup

grub-syslinux2cfg	Преобразует файл конфигурации syslinux в формат grub.cfg

8.66. Gzip-1.13

Пакет Gzip содержит программы для сжатия и распаковки файлов.

Приблизительное время сборки:	0.3 SBU
Требуемое дисковое пространство:	21 MB

8.66.1. Установка пакета Gzip

Подготовьте Gzip к компиляции:

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

Скомпилируйте пакет:

```
make
```

Чтобы протестировать пакет, выполните:

```
make check
```

Установите пакет:

```
make install
```

8.66.2. Содержимое пакета Gzip

Установленные программы:	gunzip, gzexe, gzip, uncompress (жесткая ссылка на gunzip), zcat, zcmp, zdiff, zegrep, zfgrep, zforce, zgrep, zless, zmore и znew

Краткое описание

gunzip	Распаковывает gzip-файлы
gzexe	Создает самораспаковывающиеся исполняемые файлы
gzip	Сжимает файлы, используя алгоритм Lempel-Ziv (LZ77).
uncompress	Распаковывает сжатые файлы
zcat	Распаковывает указанные сжатые файлы в стандартный поток вывода
zcmp	Запускает **cmp** для архивированных файлов
zdiff	Запускает **diff** для архивированных файлов
zegrep	Запускает **egrep** для архивированных файлов
zfgrep	Запускает **fgrep** для архивированных файлов
zforce	Принудительно устанавливает расширение `.gz` всем сжатым файлам, чтобы **gzip** не сжимал их снова; это может быть полезно, когда имена файлов были обрезаны во время передачи файла
zgrep	Запускает **grep** для архивированных файлов
zless	Запускает **less** для архивированных файлов
zmore	Запускает **more** для архивированных файлов
znew	Повторно сжимает файлы из формата **compress** в формат **gzip** — из `.Z` в `.gz`

8.67. IPRoute2-6.10.0

Пакет IPRoute2 содержит набор программ для базового и расширенного администрирования сетей IPv4.

Приблизительное время сборки: 0.1 SBU
Требуемое дисковое пространство: 17 MB

8.67.1. Установка пакета IPRoute2

Программа **arpd**, входящая в этот пакет, не будет собрана, поскольку зависит от Berkeley DB, которая не установлена в LFS. Однако каталог и справочная страница для **arpd** все равно будут установлены. Предотвратить это можно, выполнив приведенные ниже команды.

```
sed -i /ARPD/d Makefile
rm -fv man/man8/arpd.8
```

Скомпилируйте пакет:

```
make NETNS_RUN_DIR=/run/netns
```

Этот пакет не содержит рабочего набора тестов.

Установите пакет:

```
make SBINDIR=/usr/sbin install
```

По желанию, установите документацию:

```
mkdir -pv             /usr/share/doc/iproute2-6.10.0
cp -v COPYING README* /usr/share/doc/iproute2-6.10.0
```

8.67.2. Содержимое пакета IPRoute2

Установленные программы: bridge, ctstat (ссылка на lnstat), genl, ifstat, ip, lnstat, nstat, routel, rtacct, rtmon, rtpr, rtstat (ссылка на lnstat), ss и tc
Созданные каталоги: /etc/iproute2, /usr/lib/tc и /usr/share/doc/iproute2-6.10.0

Краткое описание

bridge Настраивает сетевые мосты

ctstat Утилита состояния подключения

genl Универсальный интерфейс утилиты netlink

ifstat Показывает статистику интерфейса, включая количество переданных и полученных пакетов по интерфейсам.

ip Основной исполняемый файл. Он имеет несколько различных функций, в том числе эти:
ip link *<device>* позволяет пользователям просматривать состояние устройств и вносить изменения
ip addr позволяет пользователям просматривать адреса и их свойства, добавлять новые адреса и удалять старые
ip neighbor позволяет пользователям просматривать связи с соседями и их свойства, добавлять новые записи и удалять старые
ip rule позволяет пользователям просматривать политики маршрутизации и изменять их
ip route позволяет пользователям просматривать таблицу маршрутизации и изменять правила таблицы маршрутизации

ip tunnel позволяет пользователям просматривать IP-туннели и их свойства, а также изменять их
ip maddr позволяет пользователям просматривать multicast адреса и их свойства и изменять их
ip mroute позволяет пользователям устанавливать, изменять или удалять multicast маршрутизацию.
ip monitor позволяет пользователям постоянно отслеживать состояние устройств, адресов и маршрутов

lnstat Предоставляет сетевую статистику Linux; это обобщенная и более полнофункциональная замена старой программы **rtstat**

nstat Отображает сетевую статистику

routel Компонент **ip route** для просмотра таблиц маршрутизации

rtacct Отображает содержимое `/proc/net/rt_acct`

rtmon Мониторит изменения таблицы маршрутизации

rtpr Преобразует вывод **ip -o** в удобочитаемую форму

rtstat Утилита состояния маршрута

ss Аналогично команде **netstat** показывает активные соединения

tc Управление трафиком для реализаций качества обслуживания (QoS) и класса обслуживания (CoS)
tc qdisc позволяет пользователям настроить дисциплину обработки очередей
tc class позволяет пользователям настраивать классы, на основе планирования дисциплины обработки очередей
tc filter позволяет пользователям настроить фильтрацию пакетов QOS/COS
tc monitor может использоваться для просмотра изменений, внесенных в управление трафиком в ядре

8.68. Kbd-2.6.4

Пакет Kbd содержит файлы таблиц клавиш, консольные шрифты и утилиты клавиатуры.

Приблизительное время сборки: 0.1 SBU

Требуемое дисковое пространство: 34 MB

8.68.1. Установка пакета Kbd

Поведение клавиш backspace и delete не согласуется между раскладками в пакете Kbd. Следующий патч исправляет эту проблему для раскладок i386:

```
patch -Np1 -i ../kbd-2.6.4-backspace-1.patch
```

После исправления клавиша backspace генерирует символ с кодом 127, а клавиша delete генерирует хорошо известную escape-последовательность.

Удалите ненужную программу **resizecons** (она требуется несуществующей svgalib для предоставления файлов видеорежима — для нормального использования **setfont**, который правильно определяет размеры консоли) вместе с ее справочной страницей.

```
sed -i '/RESIZECONS_PROGS=/s/yes/no/' configure
sed -i 's/resizecons.8 //' docs/man/man8/Makefile.in
```

Подготовьте Kbd для компиляции:

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

Значение параметра configure:

`--disable-vlock`
: Этот параметр предотвращает сборку утилиты vlock, поскольку для неё требуется библиотека PAM, которая недоступна в среде chroot.

Скомпилируйте пакет:

```
make
```

Чтобы протестировать пакет, выполните:

```
make check
```

Установите пакет:

```
make install
```

Примечание

Для некоторых языков (например белорусского) пакет Kbd не предоставляет подходящую раскладку, штатная раскладка «by» предполагает кодировку ISO-8859-5, а обычно используется раскладка CP1251. Пользователи таких языков должны отдельно загрузить рабочую раскладку.

По желанию, установите документацию:

```
cp -R -v docs/doc -T /usr/share/doc/kbd-2.6.4
```

8.68.2. Содержимое пакета Kbd

Установленные программы:	chvt, deallocvt, dumpkeys, fgconsole, getkeycodes, kbdinfo, kbd_mode, kbdrate, loadkeys, loadunimap, mapscrn, openvt, psfaddtable (ссылка на psfxtable), psfgettable (ссылка на psfxtable), psfstriptable (ссылка на psfxtable), psfxtable, setfont, setkeycodes, setleds, setmetamode, setvtrgb, showconsolefont, showkey, unicode_start и unicode_stop
Созданные каталоги:	/usr/share/consolefonts, /usr/share/consoletrans, /usr/share/keymaps, /usr/share/doc/kbd-2.6.4 и /usr/share/unimaps

Краткое описание

chvt	Изменяет используемый виртуальный терминал
deallocvt	Освобождает неиспользуемые виртуальные терминалы
dumpkeys	Создает дамп таблиц перевода клавиатуры
fgconsole	Выводит номер активного виртуального терминала
getkeycodes	Выводит таблицу ядра соответствия сканкода и кода клавиши
kbdinfo	Получает информацию о состоянии консоли
kbd_mode	Выводит или устанавливает режим клавиатуры
kbdrate	Устанавливает частоту повторных нажатий клавиш и задержки клавиатуры
loadkeys	Загружает таблицу преобразования клавиатуры
loadunimap	Загружает таблицу ядра отображения символов юникода
mapscrn	Устаревшая программа, которая использовалась для загрузки определяемой пользователем таблицы соответствия выводимых символов в драйвер консоли; теперь эту функцию выполняет **setfont**
openvt	Запускает программу на новом виртуальном терминале (VT)
psfaddtable	Добавляет таблицу символов Unicode в консольный шрифт.
psfgettable	Извлекает встроенную таблицу символов Unicode из консольного шрифта.
psfstriptable	Удаляет встроенную таблицу символов Unicode из консольного шрифта.
psfxtable	Обрабатывает таблицы символов Unicode для консольных шрифтов.
setfont	Изменяет шрифты Enhanced Graphic Adapter (EGA) и Video Graphics Array (VGA), используемые в консоли
setkeycodes	Загружает таблицу соответствия сканкодов ядра и кодов клавиш; это удобно, если на клавиатуре есть нестандартные клавиши
setleds	Устанавливает значения флагов клавиатуры и индикаторов (обычно - светодиоды)
setmetamode	Определяет обработку метаклавиши на клавиатуре (обычно, это клавиша Win)
setvtrgb	Устанавливает цветовую схему консоли для всех виртуальных терминалов
showconsolefont	Показывает текущий шрифт экрана консоли EGA/VGA
showkey	Показывает сканкоды, код клавиши и код ASCII для клавиш, нажатых на клавиатуре
unicode_start	Переводит клавиатуру и консоль в режим UNICODE. [Не используйте эту программу, если вы не используете файл раскладки для кодировки ISO-8859-1. Для других кодировок эта утилита выдает неправильные результаты].
unicode_stop	Возвращает клавиатуру и консоль из режима UNICODE

8.69. Libpipeline-1.5.7

Пакет Libpipeline содержит библиотеку для гибкого и удобного управления подпроцессами.

Приблизительное время сборки:	0.1 SBU
Требуемое дисковое пространство:	9.7 MB

8.69.1. Установка пакета Libpipeline

Подготовьте Libpipeline к компиляции:

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

Скомпилируйте пакет:

```
make
```

Чтобы протестировать пакет, выполните:

```
make check
```

Установите пакет:

```
make install
```

8.69.2. Содержимое пакета Libpipeline

Установленные библиотеки:	libpipeline.so

Краткое описание

`libpipeline`	Эта библиотека используется для безопасного построения конвейеров между подпроцессами.

8.70. Make-4.4.1

Пакет Make содержит программу, управляющую генерацией исполняемых и других файлов, из исходного кода.

Приблизительное время сборки: 0.7 SBU
Требуемое дисковое пространство: 13 MB

8.70.1. Установка пакета Make

Подготовьте Make к компиляции:

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

Скомпилируйте пакет:

```
make
```

Чтобы протестировать пакет, выполните:

```
chown -R tester .
su tester -c "PATH=$PATH make check"
```

Установите пакет:

```
make install
```

8.70.2. Содержимое пакета Make

Установленные программы: make

Краткое описание

make Автоматически определяет, какие части пакета необходимо (пере)компилировать и запускает соответствующие команды.

8.71. Patch-2.7.6

Пакет Patch содержит программу для изменения или создания файлов путём наложение «патча», обычно, создаваемого программой **diff**.

Приблизительное время сборки: 0.2 SBU
Требуемое дисковое пространство: 12 MB

8.71.1. Установка пакета Patch

Подготовьте Patch к компиляции:

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

Скомпилируйте пакет:

```
make
```

Чтобы протестировать пакет, выполните:

```
make check
```

Установите пакет:

```
make install
```

8.71.2. Содержимое пакета Patch

Установленные программы: patch

Краткое описание

patch Изменяет файлы в соответствии с файлом исправления (патч обычно представляет собой список отличий, создаваемый с помощью программы **diff**. Применяя их к исходным файлам, **patch** создает исправленные версии.)

8.72. Tar-1.35

Пакет Tar предоставляет возможность создавать tar архивы, а также производить с ними различные манипуляции. Tar может распаковать предварительно созданный архив, добавить или обновить файлы в нём, вернуть список файлов в архиве.

Приблизительное время сборки: 0.7 SBU
Требуемое дисковое пространство: 43 MB

8.72.1. Установка пакета Tar

Подготовьте Tar к компиляции:

```
FORCE_UNSAFE_CONFIGURE=1  \
./configure --prefix=/usr
```

Значение параметра configure:

`FORCE_UNSAFE_CONFIGURE=1`
Этот параметр принудительно запускает тест для `mknod` от имени пользователя `root`. Обычно считается опасным запускать этот тест от имени пользователя `root`, но, поскольку он выполняется в системе, которая была собрана лишь частично, его переопределение допустимо.

Скомпилируйте пакет:

```
make
```

Чтобы протестировать пакет, выполните:

```
make check
```

Известно, что один тест, capabilities: binary store/restore, завершается ошибкой при запуске, потому что в LFS отсутствует selinux, он будет пропущен, если ядро хоста не поддерживает расширенные атрибуты или метки безопасности файловой системы, используемой для сборки LFS.

Установите пакет:

```
make install
make -C doc install-html docdir=/usr/share/doc/tar-1.35
```

8.72.2. Содержимое пакета Tar

Установленные программы: tar
Созданные каталоги: /usr/share/doc/tar-1.35

Краткое описание

tar Создает архивы, извлекает файлы и отображает содержимое архивов, также известных как Тарболл.

8.73. Texinfo-7.1

Пакет Texinfo содержит программы для чтения, записи и преобразования информационных страниц.

Приблизительное время сборки: 0.3 SBU
Требуемое дисковое пространство: 139 MB

8.73.1. Установка пакета Texinfo

Подготовьте Texinfo к компиляции:

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

Скомпилируйте пакет:

```
make
```

Чтобы протестировать пакет, выполните:

```
make check
```

Установите пакет:

```
make install
```

По желанию установите компоненты, входящие в пакет TeX:

```
make TEXMF=/usr/share/texmf install-tex
```

Значение параметра make:

TEXMF=/usr/share/texmf
Переменная makefile TEXMF содержит расположение корня дерева TeX, это понадобится, если, например, пакет TeX планируется установить позже.

Система документации использует простой текстовый файл для хранения списка пунктов меню. Файл находится в /usr/share/info/dir . К сожалению, из-за случайных проблем в Makefile различных пакетов он иногда может не синхронизироваться с информационными страницами, установленными в системе. Если когда-либо потребуется пересоздать файл /usr/share/info/dir , следующие необязательные команды решают эту задачу:

```
pushd /usr/share/info
  rm -v dir
  for f in *
    do install-info $f dir 2>/dev/null
  done
popd
```

8.73.2. Содержимое пакета Texinfo

Установленные программы: info, install-info, makeinfo (ссылка на texi2any), pdftexi2dvi, pod2texi, texi2any, texi2dvi, texi2pdf, и texindex
Установленные библиотеки: MiscXS.so, Parsetexi.so и XSParagraph.so (все в /usr/lib/texinfo)
Созданные каталоги: /usr/share/texinfo и /usr/lib/texinfo

Краткое описание

info Используется для чтения информационных страниц, которые похожи на справочные страницы, но гораздо подробнее описывают применение всех доступных параметров командной строки [Например, сравните **man bison** и **info bison**.]

install-info	Используется для установки информационных страниц; он обновляет записи в индексном файле команды **info**
makeinfo	Переводит исходные документы Texinfo в информационные страницы, обычный текст или HTML.
pdftexi2dvi	Используется для форматирования документа Texinfo в файл Portable Document Format (PDF).
pod2texi	Преобразует Pod в формат Texinfo
texi2any	Переводит исходную документацию Texinfo в различные другие форматы.
texi2dvi	Используется для форматирования документа Texinfo в независимый от устройства файл, который можно распечатать
texi2pdf	Используется для форматирования данного документа Texinfo в файл Portable Document Format (PDF).
texindex	Используется для сортировки индексных файлов Texinfo.

8.74. Vim-9.1.0660

Пакет Vim содержит мощный текстовый редактор.

Приблизительное время сборки:	2.9 SBU
Требуемое дисковое пространство:	245 MB

Альтернативы Vim

Если вы предпочитаете другой текстовый редактор, например, Emacs, Joe или Nano, обратитесь к *https://mirror.linuxfromscratch.ru/blfs/view/12.2/postlfs/editors.html* за рекомендациями по установке.

8.74.1. Установка пакета Vim

Во-первых, измените расположение файла конфигурации `vimrc` на `/etc` :

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

Подготовьте Vim к компиляции:

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

Скомпилируйте пакет:

```
make
```

Чтобы подготовить тесты, убедитесь, что пользователь `tester` может писать в исходное дерево:

```
chown -R tester .
```

Теперь запустите тесты от имени пользователя `tester`:

```
su tester -c "TERM=xterm-256color LANG=en_US.UTF-8 make -j1 test" \
   &> vim-test.log
```

Набор тестов выводит на экран много двоичных данных. Это может вызвать проблемы с настройками текущего терминала (особенно, когда мы переопределяем переменную `TERM`, чтобы удовлетворить некоторые требования набора тестов). Чтобы этого избежать, перенаправьте вывод в файл журнала, как показано выше. Тест пройден успешно, если лог по завершении содержит текст: `ALL DONE`.

Установите пакет:

```
make install
```

Многие пользователи рефлекторно набирают **vi** вместо **vim**. Чтобы разрешить выполнение **vim**, когда пользователи вводят **vi**, создайте символическую ссылку как для двоичного файла, так и для справочной страницы:

```
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
```

По умолчанию документация Vim установливается в каталог `/usr/share/vim` . Следующая символическая ссылка позволяет получить доступ к документации через каталог `/usr/share/doc/vim-9.1.0660` , что согласуется с расположением документации остальных пакетов:

```
ln -sv ../vim/vim91/doc /usr/share/doc/vim-9.1.0660
```

Если в LFS будет установлена система X Window, может потребоваться перекомпилировать Vim после установки X. Vim поставляется с графической версией редактора, для которой требуется установка X и некоторых дополнительных библиотек. Для получения дополнительной информации об этом процессе обратитесь к документации по Vim и странице установки Vim в книге BLFS по адресу *https://mirror.linuxfromscratch.ru/blfs/view/12.2/postlfs/vim.html*.

8.74.2. Настройка Vim

По умолчанию **vim** работает в режиме, несовместимом с vi. Это может показаться необычным для пользователей, которые в прошлом использовали другие редакторы. Параметр «nocompatible» включен ниже, чтобы подчеркнуть тот факт, что используется новое поведение. Настройка также напоминает тем, кто хотел бы перейти в режим «compatible», что параметр должен быть первым в файле конфигурации. Это необходимо, потому что изменяются другие параметры, и переопределения происходят после этой настройки. Создайте файл конфигурации **vim** по умолчанию, выполнив следующие действия:

```
cat > /etc/vimrc << "EOF"
" Begin /etc/vimrc

" Ensure defaults are set before customizing settings, not after
source $VIMRUNTIME/defaults.vim
let skip_defaults_vim=1

set nocompatible
set backspace=2
set mouse=
syntax on
if (&term == "xterm") || (&term == "putty")
  set background=dark
endif

" End /etc/vimrc
EOF
```

Параметр *set nocompatible* заставляет **vim** вести себя более правильно (по умолчанию), чем vi-совместимый способ. Удалите «no», чтобы сохранить старое поведение **vi**. Параметр *set backspace=2* позволяет удалять символы через перенос строки, автоматические отступы и начало вставки. Параметр *syntax on* включает подсветку синтаксиса vim. Параметр *set mouse=* позволяет правильно вставлять текст с помощью мыши при работе в chroot или через удаленное соединение. Наконец, оператор *if* с параметром *set background=dark* корректирует предположение **vim** о цвете фона некоторых эмуляторов терминала. Это придает подсветке лучшую цветовую схему для использования на черном фоне этих программ.

Документацию по другим доступным параметрам можно получить, выполнив следующую команду:

```
vim -c ':options'
```

Примечание

По умолчанию vim устанавливает файлы проверки орфографии только для английского языка. Для установки файлов проверки орфографии других языков, скопируйте файлы `.spl` и, при необходимости, `.sug` для вашего языка и кодировки символов из `runtime/spell`, сохраните их в `/usr/share/vim/vim91/spell/`.

Чтобы использовать эти файлы проверки орфографии, необходимо указать параметры для vim в файле `/etc/vimrc`, пример:

```
set spelllang=en,ru
set spell
```

Дополнительные сведения смотрите в файле `runtime/spell/README.txt`.

8.74.3. Содержимое пакета Vim

Установленные программы:	ex (ссылка на vim), rview (ссылка на vim), rvim (ссылка на vim), vi (ссылка на vim), view (ссылка на vim), vim, vimdiff (ссылка на vim), vimtutor и xxd
Созданные каталоги:	/usr/share/vim

Краткое описание

ex	Запускает **vim** в режиме ex
rview	Это ограниченная версия **view**; никакие команды оболочки не могут быть запущены, и **view** не может быть приостановлен
rvim	Это ограниченная версия **vim**; никакие команды оболочки не могут быть запущены, и **vim** не может быть приостановлен
vi	Ссылка на **vim**
view	Запускает **vim** в режиме только для чтения
vim	Сам редактор
vimdiff	Редактирует две или три версии файла с помощью **vim** и показывает различия
vimtutor	Обучает основным горячим клавишам и командам **vim**
xxd	Создает шестнадцатеричный дамп данного файла; он также может выполнять обратную операцию, поэтому его можно использовать для бинарных патчей

8.75. MarkupSafe-2.1.5

MarkupSafe — это модуль Python, реализующий безопасное использование строк в языках разметки XML/HTML/XHTML

Приблизительное время сборки: менее 0.1 SBU
Требуемое дисковое пространство: 508 KB

8.75.1. Установка пакета MarkupSafe

Скомпилируйте MarkupSafe с помощью следующей команды:

```
pip3 wheel -w dist --no-cache-dir --no-build-isolation --no-deps $PWD
```

С этим пакетом не поставляется тестов.

Установите пакет:

```
pip3 install --no-index --no-user --find-links dist Markupsafe
```

8.75.2. Содержимое пакета MarkupSafe

Созданные каталоги: /usr/lib/python3.12/site-packages/MarkupSafe-2.1.5.dist-info

8.76. Jinja2-3.1.4

Jinja2 - это модуль Python, который реализует простой язык шаблонов pythonic

Приблизительное время сборки: менее 0.1 SBU
Требуемое дисковое пространство: 2.5 MB

8.76.1. Установка пакета Jinja2

Соберите пакет:

```
pip3 wheel -w dist --no-cache-dir --no-build-isolation --no-deps $PWD
```

Установите пакет:

```
pip3 install --no-index --no-user --find-links dist Jinja2
```

8.76.2. Содержимое пакета Jinja2

Созданные каталоги: /usr/lib/python3.12/site-packages/Jinja2-3.1.4.dist-info

8.77. Udev из Systemd-256.4

Пакет Udev содержит программы для динамического создания узлов устройств.

Приблизительное время сборки:	0.2 SBU
Требуемое дисковое пространство:	144 MB

8.77.1. Установка пакета Udev

Udev является частью пакета systemd-256.4. Используйте файл systemd-256.4.tar.xz в качестве исходного архива.

Удалите две ненужные группы `render` и `sgx`, из правил udev по умолчанию:

```
sed -i -e 's/GROUP="render"/GROUP="video"/' \
       -e 's/GROUP="sgx", //' rules.d/50-udev-default.rules.in
```

Удалите одно правило udev, требующее полной установки Systemd:

```
sed '/systemd-sysctl/s/^/#/' -i rules.d/99-systemd.rules.in
```

Измените жестко заданные пути к файлам конфигурации сети для автономной установки udev:

```
sed '/NETWORK_DIRS/s/systemd/udev/' -i src/basic/path-lookup.h
```

Подготовьте Udev к компиляции:

```
mkdir -p build
cd       build

meson setup ..                  \
      --prefix=/usr             \
      --buildtype=release       \
      -D mode=release           \
      -D dev-kvm-mode=0660      \
      -D link-udev-shared=false \
      -D logind=false           \
      -D vconsole=false
```

Значение параметров meson:

`--buildtype=release`
: Этот параметр переопределяет тип сборки по умолчанию («debug»), который создает неоптимизированные двоичные файлы.

`-D mode=release`
: Отключает некоторые функции, которые разработчики считают экспериментальными.

`-D dev-kvm-mode=0660`
: По умолчанию правило udev разрешает всем пользователям доступ к `/dev/kvm` . Редакторы LFS считают это опасным. Данная опция переопределяет разрешение по умолчанию.

`-D link-udev-shared=false`
: Эта опция запрещает udev связываться с внутренней общей библиотекой systemd `libsystemd-shared` . Эта библиотека предназначена для совместного использования многими компонентами Systemd, и ее использование избыточно, когда установлен только udev.

`-D logind=false -D vconsole=false`
: Эти параметры предотвращают создание нескольких файлов правил udev, принадлежащих другим компонентам Systemd, которые мы не будем устанавливать.

Получите список предоставляемых udev helpers и сохраните его в переменной (экспортировать его не обязательно, но это упрощает сборку от имени обычного пользователя или при использовании менеджера пакетов):

```
export udev_helpers=$(grep "'name' :" ../src/udev/meson.build | \
                      awk '{print $3}' | tr -d ",'" | grep -v 'udevadm')
```

Соберите только компоненты, необходимые для udev:

```
ninja udevadm systemd-hwdb \
      $(ninja -n | grep -Eo '(src/(lib)?udev|rules.d|hwdb.d)/[^ ]*') \
      $(realpath libudev.so --relative-to .)                          \
      $udev_helpers
```

Установите пакет:

```
install -vm755 -d {/usr/lib,/etc}/udev/{hwdb.d,rules.d,network}
install -vm755 -d /usr/{lib,share}/pkgconfig
install -vm755 udevadm                    /usr/bin/
install -vm755 systemd-hwdb               /usr/bin/udev-hwdb
ln      -svfn  ../bin/udevadm             /usr/sbin/udevd
cp      -av    libudev.so{,*[0-9]}        /usr/lib/
install -vm644 ../src/libudev/libudev.h   /usr/include/
install -vm644 src/libudev/*.pc           /usr/lib/pkgconfig/
install -vm644 src/udev/*.pc              /usr/share/pkgconfig/
install -vm644 ../src/udev/udev.conf      /etc/udev/
install -vm644 rules.d/* ../rules.d/README         /usr/lib/udev/rules.d/
install -vm644 $(find ../rules.d/*.rules \
                      -not -name '*power-switch*') /usr/lib/udev/rules.d/
install -vm644 hwdb.d/*  ../hwdb.d/{*.hwdb,README} /usr/lib/udev/hwdb.d/
install -vm755 $udev_helpers                       /usr/lib/udev
install -vm644 ../network/99-default.link          /usr/lib/udev/network
```

Установите некоторые пользовательские правила и вспомогательные файлы, полезные в среде LFS:

```
tar -xvf ../../udev-lfs-20230818.tar.xz
make -f udev-lfs-20230818/Makefile.lfs install
```

Установите справочные страницы:

```
tar -xf ../../systemd-man-pages-256.4.tar.xz                            \
    --no-same-owner --strip-components=1                                \
    -C /usr/share/man --wildcards '*/udev*' '*/libudev*'                \
                                  '*/systemd.link.5'                    \
                                  '*/systemd-'{hwdb,udevd.service}.8

sed 's|systemd/network|udev/network|'                                   \
    /usr/share/man/man5/systemd.link.5                                  \
  > /usr/share/man/man5/udev.link.5

sed 's/systemd\(\\?-\)/udev\1/' /usr/share/man/man8/systemd-hwdb.8   \
                             > /usr/share/man/man8/udev-hwdb.8

sed 's|lib.*udevd|sbin/udevd|'                                          \
    /usr/share/man/man8/systemd-udevd.service.8                         \
  > /usr/share/man/man8/udevd.8

rm /usr/share/man/man*/systemd*
```

Наконец, сбросьте значение переменной `udev_helpers` :

```
unset udev_helpers
```

8.77.2. Установка Udev - 32-битная версия

Выполните очистку после предыдущей сборки (находясь в каталоге сборки):

```
rm -rf *
```

Подготовьте Udev к компиляции:

```
PKG_CONFIG_PATH="/usr/lib32/pkgconfig" \
CC="gcc -m32 -march=i686"              \
CXX="g++ -m32 -march=i686"             \
LANG=en_US.UTF-8                       \
meson setup \
      --prefix=/usr                 \
      --buildtype=release           \
      -Dmode=release                \
      -Ddev-kvm-mode=0660           \
      -Dlink-udev-shared=false      \
      -Dlogind=false                \
      -Dvconsole=false              \
      ..
```

Скомпилируйте пакет:

```
ninja \
      $(grep -o -E "^build (src/libudev|src/udev)[^:]*" \
        build.ninja | awk '{ print $2 }')                \
      $(realpath libudev.so --relative-to .)
```

Установите пакет:

```
mkdir -pv /usr/lib32/pkgconfig &&
cp -av libudev.so{,*[0-9]} /usr/lib32/ &&
sed -e "s;/usr/lib;&32;g" src/libudev/libudev.pc > /usr/lib32/pkgconfig/libudev.pc
```

8.77.3. Настройка Udev

Информация об аппаратных устройствах хранится в каталогах `/etc/udev/hwdb.d` и `/usr/lib/udev/hwdb.d`. Udev необходимо, чтобы эта информация была скомпилирована в двоичную базу данных `/etc/udev/hwdb.bin`. Создайте исходную базу данных:

```
udev-hwdb update
```

Эту команду необходимо запускать каждый раз при обновлении информации об оборудовании.

8.77.4. Содержимое пакета Udev

Установленные программы:	udevadm, udevd (символическая ссылка на udevadm) и udev-hwdb
Установленные библиотеки:	libudev.so
Созданные каталоги:	/etc/udev и /usr/lib/udev

Short Descriptions

udevadm	Универсальный инструмент администрирования udev: управляет демоном udevd, предоставляет информацию из базы данных Udev, отслеживает uevents, ожидает завершения uevents, проверяет конфигурацию Udev и запускает uevents для данного устройства
udevd	Демон, который прослушивает uevents в сокете netlink, создает устройства и запускает настроенные внешние программы в ответ на эти uevents
udev-hwdb	Обновляет или запрашивает базу данных оборудования
`libudev`	Библиотека для получения информации об устройствах udev
`/etc/udev`	Содержит файлы конфигурации Udev, разрешения для устройств и правила именования устройств

8.78. Man-DB-2.12.1

Пакет Man-DB содержит программы для поиска и просмотра справочных страниц.

Приблизительное время сборки: 0.3 SBU
Требуемое дисковое пространство: 43 MB

8.78.1. Установка пакета Man-DB

Подготовьте Man-DB к компиляции:

```
./configure --prefix=/usr                         \
            --docdir=/usr/share/doc/man-db-2.12.1 \
            --sysconfdir=/etc                     \
            --disable-setuid                      \
            --enable-cache-owner=bin              \
            --with-browser=/usr/bin/lynx          \
            --with-vgrind=/usr/bin/vgrind         \
            --with-grap=/usr/bin/grap             \
            --with-systemdtmpfilesdir=            \
            --with-systemdsystemunitdir=
```

Значение параметров настройки:

`--disable-setuid`
Отключает установку setuid пользователю `man` при сборке программы **man**.

`--enable-cache-owner=bin`
Изменяет владельца файлов общесистемного кэша на пользователя `bin`.

`--with-...`
Эти три аргумента используются для настройки программ по умолчанию. **lynx** текстовый веб-браузер (см. инструкции по установке в книге BLFS), **vgrind** преобразует исходные коды программ во входные данные Groff, **grap** удобен для набора графов в документах Groff. Программы **vgrind** и **grap** обычно не нужны для просмотра справочных страниц. Они не входят в состав книг LFS или BLFS, но вы можете установить их самостоятельно после сборки LFS.

`--with-systemd...`
Эти параметры предотвращают установку ненужных каталогов и файлов systemd.

Скомпилируйте пакет:

```
make
```

Чтобы протестировать пакет, выполните:

```
make check
```

Установите пакет:

```
make install
```

8.78.2. Не англоязычные страницы руководств в LFS

В следующей таблице приведены наборы символов, в которых могут быть закодированы страницы руководств пакета Man-DB, устанавливаемые в директории `/usr/share/man/<ll>` . Кроме этого, Man-DB правильно определяет, имеют ли справочные страницы, установленные в этом каталоге, кодировку UTF-8.

Таблица 8.1. Допустимые кодировки старых 8-битных страниц руководств

Язык (код)	Кодировка	Язык (код)	Кодировка
Датский (da)	ISO-8859-1	Хорватский (hr)	ISO-8859-2
Немецкий (de)	ISO-8859-1	Венгерский (hu)	ISO-8859-2
Английский (en)	ISO-8859-1	Японский (ja)	EUC-JP
Испанский (es)	ISO-8859-1	Корейский (ko)	EUC-KR
Эстонский (et)	ISO-8859-1	Литовский (lt)	ISO-8859-13
Финский (fi)	ISO-8859-1	Латышский (lv)	ISO-8859-13
Французский (fr)	ISO-8859-1	Македонский (mk)	ISO-8859-5
Ирландский (ga)	ISO-8859-1	Польский (pl)	ISO-8859-2
Галисийский (gl)	ISO-8859-1	Румынский (ro)	ISO-8859-2
Индонезийский (id)	ISO-8859-1	Русский (ru)	KOI8-R
Исландский (is)	ISO-8859-1	Словацкий (sk)	ISO-8859-2
Итальянский (it)	ISO-8859-1	Словенский (sl)	ISO-8859-2
Норвежский букмол (nb)	ISO-8859-1	Сербский латинский (sr@latin)	ISO-8859-2
Голландский (nl)	ISO-8859-1	Сербский (sr)	ISO-8859-5
Норвежский нюнорск (nn)	ISO-8859-1	Турецкий (tr)	ISO-8859-9
Норвежский (no)	ISO-8859-1	Украинский (uk)	KOI8-U
Португальский (pt)	ISO-8859-1	Вьетнамский (vi)	TCVN5712-1
Шведский (sv)	ISO-8859-1	Упрощенный китайский (zh_CN)	GBK
Белорусский (be)	CP1251	Упрощенный китайский, Сингапур (zh_SG)	GBK
Болгарский (bg)	CP1251	Традиционный китайский, Гонконг (zh_HK)	BIG5HKSCS
Чешский (cs)	ISO-8859-2	Традиционный китайский (zh_TW)	BIG5
Греческий (el)	ISO-8859-7		

Примечание

Страницы руководств на языках, которые не указаны в списке, не поддерживаются.

8.78.3. Содержимое пакета Man-DB

Установленные программы: accessdb, apropos (ссылка на whatis), catman, lexgrog, man, man-recode, mandb, manpath, и whatis
Установленные библиотеки: libman.so и libmandb.so (обе в /usr/lib/man-db)
Созданные каталоги: /usr/lib/man-db, /usr/libexec/man-db и /usr/share/doc/man-db-2.12.1

Краткое описание

accessdb Выводит содержимое базы данных **whatis** в удобочитаемой форме.

apropos	Выполняет поиск в базе данных **whatis** и отображает краткое описание системных команд, содержащих заданную строку
catman	Создает или обновляет предварительно отформатированные страницы руководств
lexgrog	Отображает однострочную сводную информацию о данной странице руководства
man	Форматирует и отображает запрошенную страницу руководства
man-recode	Преобразует страницы руководства в другую кодировку
mandb	Создает или обновляет базу данных **whatis**
manpath	Отображает содержимое переменной $MANPATH или (если переменная $MANPATH не установлена) соответствующий путь поиска, определяемый в настройках man.conf и в пользовательском окружении
whatis	Выполняет поиск в базе данных **whatis** и отображает краткие описания системных команд, в которых в описании ключей указано искомое слово
`libman`	Включает поддержку **man** во время выполнения
`libmandb`	Включает поддержку **man** во время выполнения

8.79. Procps-ng-4.0.4

Пакет Procps-ng содержит программы для мониторинга процессов.

Приблизительное время сборки: 0.1 SBU
Требуемое дисковое пространство: 26 MB

8.79.1. Установка пакета Procps-ng

Подготовьте Procps-ng к компиляции:

```
./configure --prefix=/usr                           \
            --docdir=/usr/share/doc/procps-ng-4.0.4 \
            --disable-static                        \
            --disable-kill
```

Значение параметра configure:

`--disable-kill`
Этот параметр отключает сборку команды **kill**; она будет установлена из пакета Util-linux.

Скомпилируйте пакет:

```
make
```

Чтобы запустить набор тестов, выполните:

```
chown -R tester .
su tester -c "PATH=$PATH make check"
```

Известно, что тест `ps with output flag bsdtime,cputime,etime,etimes` завершается неудачно, если ядро хоста собрано с выключенным параметром `CONFIG_BSD_PROCESS_ACCT`

Установите пакет:

```
make install
```

8.79.2. Содержимое пакета Procps-ng

Установленные программы: free, pgrep, pidof, pkill, pmap, ps, pwdx, slabtop, sysctl, tload, top, uptime, vmstat, w и watch
Установленные библиотеки: libproc-2.so
Созданные каталоги: /usr/include/procps и /usr/share/doc/procps-ng-4.0.4

Краткое описание

free Сообщает объем свободной и используемой памяти (как физической, так и файла подкачки) в системе.

pgrep Выполняет поиск процессов на основе их имени и других атрибутов

pidof Сообщает PIDы указанных программ

pkill Отправка сигналов процессам на основе их имени и других атрибутов

pmap Команда выводит детальную информацию об использование оперативной памяти процессами

ps Список запущенных процессов

pwdx	Сообщает текущий рабочий каталог процесса
slabtop	Отображает подробную информацию о кэш-памяти ядра в режиме реального времени.
sysctl	Изменяет параметры ядра во время выполнения
tload	Выводит график текущей средней загрузки системы
top	Отображает список процессов, наиболее интенсивно использующих ЦП; обеспечивает просмотр активности процессора в режиме реального времени
uptime	Сообщает сколько времени работает система, сколько пользователей вошли в систему и средние значения загрузки системы.
vmstat	Сообщает статистику виртуальной памяти, содержащую информацию о процессах, памяти, подкачке, блочном вводе/выводе (IO), прерываниях и активности ЦП.
w	Показывает, какие пользователи в настоящее время вошли в систему и с какого момента
watch	Выполняет заданную команду повторно, отображая первый экран, заполненный ее выводом; это позволяет пользователю наблюдать за изменениями с течением времени
`libproc-2`	Содержит функции, используемые большинством программ в этом пакете.

8.80. Util-linux-2.40.2

Пакет Util-linux содержит различные служебные программы. Среди них утилиты для работы с файловыми системами, консолями, разделами и сообщениями.

Приблизительное время сборки: 0.5 SBU
Требуемое дисковое пространство: 315 MB

8.80.1. Установка пакета Util-linux

Подготовьте Util-linux к компиляции:

```
./configure --bindir=/usr/bin     \
            --libdir=/usr/lib     \
            --runstatedir=/run    \
            --sbindir=/usr/sbin   \
            --disable-chfn-chsh   \
            --disable-login       \
            --disable-nologin     \
            --disable-su          \
            --disable-setpriv     \
            --disable-runuser     \
            --disable-pylibmount  \
            --disable-liblastlog2 \
            --disable-static      \
            --without-python      \
            --without-systemd     \
            --without-systemdsystemunitdir        \
            ADJTIME_PATH=/var/lib/hwclock/adjtime \
            --docdir=/usr/share/doc/util-linux-2.40.2
```

Параметры --disable и --without предотвращают появление предупреждений о сборке компонентов, для которых требуются пакеты, отсутствующие в LFS, или которые несовместимы с программами, установленными другими пакетами.

Скомпилируйте пакет:

```
make
```

По желанию создайте фиктивный файл `/etc/fstab` для выполнения двух тестов и запустите набор тестов от имени пользователя без полномочий `root`:

Предупреждение

Запуск набора тестов от имени пользователя `root` может повредить вашу систему. Чтобы запустить тесты, опция CONFIG_SCSI_DEBUG для ядра должна быть доступна в текущей работающей системе и должна быть собрана как модуль. Включение её в ядро будет прерывать загрузку. Для полного охвата тестами в систему необходимо установить другие пакеты из BLFS. По желанию, этот тест можно запустить после загрузки в готовую систему LFS:

```
bash tests/run.sh --srcdir=$PWD --builddir=$PWD
```

```
touch /etc/fstab
chown -R tester .
su tester -c "make -k check"
```

Тесты с *жесткими ссылками* завершатся неудачей, если в ядре хоста не включена опция `CONFIG_CRYPTO_USER_API_HASH` или не включено никаких опций, обеспечивающих реализацию SHA256 (например, `CONFIG_CRYPTO_SHA256` или `CONFIG_CRYPTO_SHA256_SSSE3`, если процессор поддерживает инструкции SSE3). Кроме того, тест lsfd: inotify завершится неудачно, если опция ядра `CONFIG_NETLINK_DIAG` не включена.

Установите пакет:

```
make install
```

8.80.2. Установка Util-linux - 32-битная версия

Выполните очистку после предыдущей сборки:

```
make distclean
```

Переименуйте инструмент, который необязательно, что будет использоваться скриптом configure, но будет сообщать о недопустимых путях для multilib-сборок.

```
mv /usr/bin/ncursesw6-config{,.tmp}
```

Подготовьте Util-linux к компиляции:

```
CC="gcc -m32" \
./configure ADJTIME_PATH=/var/lib/hwclock/adjtime   \
            --host=i686-pc-linux-gnu \
            --libdir=/usr/lib32      \
            --runstatedir=/run       \
            --sbindir=/usr/sbin      \
            --docdir=/usr/share/doc/util-linux-2.40.2 \
            --disable-chfn-chsh      \
            --disable-login          \
            --disable-nologin        \
            --disable-su             \
            --disable-setpriv        \
            --disable-runuser        \
            --disable-pylibmount     \
            --disable-liblastlog2    \
            --disable-static         \
            --without-python         \
            --without-systemd        \
            --without-systemdsystemunitdir
```

Восстановите ранее переименованный инструмент:

```
mv /usr/bin/ncursesw6-config{.tmp,}
```

Скомпилируйте пакет:

```
make
```

Установите пакет:

```
make DESTDIR=$PWD/DESTDIR install
cp -Rv DESTDIR/usr/lib32/* /usr/lib32
rm -rf DESTDIR
```

8.80.3. Содержимое пакета Util-linux

Установленные программы: addpart, agetty, blkdiscard, blkid, blkzone, blockdev, cal, cfdisk, chcpu, chmem, choom, chrt, col, colcrt, colrm, column, ctrlaltdel, delpart, dmesg, eject, fallocate, fdisk, fincore, findfs, findmnt, flock, fsck, fsck.cramfs, fsck.minix, fsfreeze, fstrim, getopt, hardlink, hexdump, hwclock, i386 (ссылка на setarch), ionice, ipcmk, ipcrm, ipcs, irqtop, isosize, kill, last, lastb (ссылка на last), ldattach, linux32 (link to setarch), linux64 (ссылка на setarch), logger, look, losetup, lsblk, lscpu, lsipc, lsirq, lsfd, lslocks, lslogins, lsmem, lsns, mcookie, mesg, mkfs, mkfs.bfs, mkfs.cramfs, mkfs.minix, mkswap, more, mount, mountpoint, namei, nsenter, partx, pivot_root, prlimit, readprofile, rename, renice, resizepart, rev, rfkill, rtcwake, script, scriptlive, scriptreplay, setarch, setsid, setterm, sfdisk, sulogin, swaplabel, swapoff, swapon, switch_root, taskset, uclampset, ul, umount, uname26 (ссылка на setarch), unshare, utmpdump, uuidd, uuidgen, uuidparse, wall, wdctl, whereis, wipefs, x86_64 (ссылка на setarch) и zramctl

Установленные библиотеки: libblkid.so, libfdisk.so, libmount.so, libsmartcols.so и libuuid.so

Созданные каталоги: /usr/include/blkid, /usr/include/libfdisk, /usr/include/libmount, /usr/include/libsmartcols, /usr/include/uuid, /usr/share/doc/util-linux-2.40.2 и /var/lib/hwclock

Краткое описание

addpart Сообщает ядру Linux о новых разделах

agetty Открывает порт tty, запрашивает имя для входа, а затем вызывает программу **login**

blkdiscard Очищает сектора на устройстве

blkid Утилита командной строки для вывода атрибутов блочного устройства

blkzone Используется для управления зонированными блочными системами хранения

blockdev Позволяет пользователям вызывать ioctl блочного устройства из командной строки

cal Отображает простой календарь

cfdisk Управляет таблицей разделов данного устройства

chcpu Изменяет состояние процессоров

chmem Настраивает память

choom Отображает и регулирует оценки OOM-killer, используемые для определения того, какой процесс следует завершить первым, когда в Linux заканчивается память

chrt Манипулирует атрибутами процесса в режиме реального времени

col Фильтрует обратные переносы строк из входного потока

colcrt Фильтрует данные, выдаваемые командой **nroff** на терминалы, у которых отсутствует ряд возможностей, например, отображение перечеркнутых символов или верхних и нижних индексов

colrm Фильтрует вывод указанных столбцов

column Форматирует заданный файл в несколько столбцов

ctrlaltdel Устанавливает для комбинации символов Ctrl+Alt+Del жесткую или мягкую перезагрузку

delpart Запрашивает у ядра Linux удаление раздела

dmesg Выводит загрузочные сообщения ядра

eject Извлекает съемный носитель

fallocate Предварительное выделение места под файл

fdisk Манипулирует таблицей разделов указанного устройства

fincore Подчитывает сколько страниц приложение хранит в памяти ядра

findfs Находит файловую систему по метке или универсальному уникальному идентификатору (UUID)

findmnt Представляет собой интерфейс командной строки к библиотеке libmount для работы с файлами mountinfo, fstab и mtab

flock Осуществляет блокировку файла, а затем выполняет команду, не снимая блокировку

fsck Используется для проверки и, при необходимости, восстановления файловых систем

fsck.cramfs Выполняет проверку целостности файловой системы Cramfs на данном устройстве

fsck.minix Выполняет проверку целостности файловой системы Minix на данном устройстве

fsfreeze Очень простая программа-обертка для выполнение операций с драйвером ядра FIFREEZE/FITHAW ioctl

fstrim Освобождает неиспользованные блоки смонтированной файловой системы

getopt Разбирает параметры указанной командной строки

hardlink Объединяет дубликаты файлов путем создания жестких ссылок

hexdump Создает дамп указанного файла в шестнадцатеричном, десятичном, восьмеричном или ascii-формате

hwclock Читает или устанавливает значение аппаратных часов системы, называемых также часами реального времени (RTC- Real-Time Clock) или часами БИОС (BIOS - Basic Input-Output System)

i386 Символьная ссылка на setarch

ionice Читает или устанавливает класс и приоритет обработки ввода/вывода для программ

ipcmk Создает различные ресурсы межпроцессного взаимодействия (IPC)

ipcrm Удаляет указанный ресурс межпроцессного взаимодействия (IPC)

ipcs Предоставляет информацию о состоянии IPC

irqtop Отображает информацию о счетчике прерываний ядра в стиле *top(1)*

isosize Сообщает о размере файловой системы iso9660

kill Посылает сигналы процессам

last Показывает, какие пользователи в последний раз входили (и выходили), выполняя поиск в файле `/var/log/wtmp` ; кроме этого показывает информацию о загрузке системы, завершение работы и изменениях уровня выполнения

lastb Показывает неудачные попытки входа в систему, зарегистрированные в `/var/log/btmp`

ldattach Назначает устройству последовательного доступа алгоритм, определяющий дисциплину обслуживания этого устройства

linux32 Символическая ссылка на setarch

linux64 Символическая ссылка на setarch

logger Добавляет указанное сообщение в системный журнал

look Отображает строки, начинающиеся с указанной последовательности символов

losetup Настраивает и управляет устройствами типа loop

lsblk	Выводит информацию обо всех или выбранных блочных устройствах в древовидном формате
lscpu	Выводит информацию об архитектуре процессора
lsfd	Отображает информацию об открытых файлах; заменяет **lsof**
lsipc	Выводит информацию об объектах IPC, которые в настоящее время используются в системе
lsirq	Отображает информацию о счетчике прерываний ядра
lslocks	Отображает список всех заблокированных в настоящее время файлов в системе
lslogins	Выводит информацию о пользователях, группах и системных учетных записях
lsmem	Отображает диапазоны доступной памяти с указанием их оперативного статуса
lsns	Отображает список пространств имен
mcookie	Генерирует для **xauth** магические куки (128-битные случайные числа в шестнадцатеричном формате)
mesg	Определяет, могут ли другие пользователи отправлять сообщения на терминал текущего пользователя
mkfs	Создает файловую систему на устройстве (обычно это раздел жесткого диска)
mkfs.bfs	Создает файловую систему Santa Cruz Operations (SCO) bfs
mkfs.cramfs	Создает файловую систему cramfs
mkfs.minix	Создает файловую систему Minix
mkswap	Инициализирует данное устройство или файл для использования в качестве области подкачки
more	Фильтр постраничного вывода текста
mount	Подключение файловой системы, находящейся на заданном устройстве, к указанному каталогу в дереве файловой системы
mountpoint	Проверяет, является ли каталог точкой монтирования
namei	Разделяет на составляющие путь к файлу или каталогу, показывая информацию о типе каждого элемента
nsenter	Запускает программу в пространстве имен других процессов
partx	Сообщает ядру информацию о наличии и количестве разделов, находящихся на диске
pivot_root	Делает данную файловую систему новой корневой файловой системой текущего процесса
prlimit	Получает и устанавливает ограничения использования ресурсов процесса
readprofile	Читает информацию о профилировании ядра
rename	Переименовывает заданные файлы, заменяя одну строку другой
renice	Изменяет приоритет запущенных процессов
resizepart	Запрашивает у ядра Linux изменение размера раздела
rev	Меняет в указанном файле порядок строк на обратный
rfkill	Внструмент командной строки для управления беспроводными устройствами
rtcwake	Используется для перехода системы в спящий режим до указанного времени пробуждения
script	Создает скрипт терминальной сессии

scriptlive	Перезапускает скрипт терминальной сессии, используя информацию о времени
scriptreplay	Воспроизводит скрипт в соответствие с указанным временем запуска
setarch	В окружении, используемом новой программой, изменяет информацию об архитектуре и устанавливает флаги персонализации
setsid	Запускает указанную программу в новом сеансе
setterm	Устанавливает атрибуты терминала
sfdisk	Управляет таблицей разделов диска
sulogin	Позволяет пользователю `root` входить в систему; обычно он вызывается **init**, когда система переходит в однопользовательский режим
swaplabel	Изменяет UUID и метку раздела подкачки
swapoff	Отключает устройства и файлы подкачки
swapon	Включает устройства и файлы, применяемые для раздела подкачки, а также выводит список устройств и файлов, используемых в данный момент
switch_root	Переключается на другую файловую систему и устанавливает её в качестве корневой
taskset	Устанавливает привязку процессора к процессу
uclampset	Управляет атрибутами ограничения использования системы или процесса
ul	Фильтр для преобразования символов подчеркивания в escape-последовательности
umount	Размонтирует файловую систему из дерева ФС
uname26	Символическая ссылка на setarch
unshare	Позволяет процессу (или потоку) отделить части своего контекста выполнения, которые используются совместно с другими процессами (или потоками)
utmpdump	Отображает содержимое указанного файла входа в систему в удобном для пользователя формате
uuidd	Демон, используемый библиотекой UUID для создания безопасных и гарантированно уникальных идентификаторов UUID
uuidgen	Создает новые идентификаторы (UUID). Каждый новый UUID - это случайная последовательность, которая, будет с очень высокой вероятностью уникальной среди всех идентификаторов, созданных как на локальной машине, так и на любых других системах, в прошлом и будущем (2^{128} или около 3,4 x 10^{38} вариантов)
uuidparse	Утилита для анализа уникальных идентификаторов
wall	Отображает содержимое файла или, по умолчанию, его вывод на терминалах всех пользователей, вошедших в систему в данный момент
wdctl	Показывает статус аппаратного сторожевого таймера
whereis	Сообщает местоположение двоичного файла, исходного кода и справочной страницы для указанной команды
wipefs	Стирает с устройства сигнатуру файловой системы
x86_64	Символическая ссылка на setarch
zramctl	Программа для настройки и управления устройствами zram (сжатый RAM-диск)
`libblkid`	Содержит подпрограммы для идентификации устройства и извлечения токена
`libfdisk`	Содержит подпрограммы для управления таблицами разделов
`libmount`	Содержит подпрограммы для монтирования и размонтирования блочных устройств

`libsmartcols`	Содержит подпрограммы для более удобного вывода на экран информации в табличном виде
`libuuid`	Содержит подпрограммы для генерации уникальных идентификаторов для объектов, которые могут быть доступны за пределами локальной системы

8.81. E2fsprogs-1.47.1

Пакет E2fsprogs содержит утилиты для работы с файловой системой `ext2`. Также он поддерживает журналируемые файловые системы `ext3` и `ext4`.

Приблизительное время сборки: 2.4 SBU на жестком диске, 0.5 SBU на SSD диске
Требуемое дисковое пространство: 98 MB

8.81.1. Установка пакета E2fsprogs

В документации к E2fsprogs рекомендуется выполнять сборку в подкаталоге папки с исходниками:

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

Подготовьте E2fsprogs к компиляции:

```
../configure --prefix=/usr           \
             --sysconfdir=/etc       \
             --enable-elf-shlibs     \
             --disable-libblkid      \
             --disable-libuuid       \
             --disable-uuidd         \
             --disable-fsck
```

Значение параметров настройки:

`--enable-elf-shlibs`
Параметр создает общие библиотеки, которые используют некоторые программы в этом пакете.

`--disable-*`
Эти параметры предотвращают сборку и установку библиотек `libuuid` и `libblkid`, демона `uuidd`, и обертку для **fsck**, поскольку util-linux устанавливает более свежие версии.

Скомпилируйте пакет:

```
make
```

Чтобы запустить тесты, выполните:

```
make check
```

Известно, что один тест, с именем `m_assume_storage_prezeroed` , завершается ошибкой.

Установите пакет:

```
make install
```

Удалите ненужные статические библиотеки:

```
rm -fv /usr/lib/{libcom_err,libe2p,libext2fs,libss}.a
```

Этот пакет устанавливает сжатый файл `.info` но не обновляет общесистемный файл `dir`. Разархивируйте этот файл, а затем обновите системный файл `dir`, используя следующие команды:

```
gunzip -v /usr/share/info/libext2fs.info.gz
install-info --dir-file=/usr/share/info/dir /usr/share/info/libext2fs.info
```

По желанию, создайте и установите дополнительную документацию, выполнив следующие команды:

```
makeinfo -o      doc/com_err.info ../lib/et/com_err.texinfo
install -v -m644 doc/com_err.info /usr/share/info
install-info --dir-file=/usr/share/info/dir /usr/share/info/com_err.info
```

8.81.2. Настройка E2fsprogs

Файл `/etc/mke2fs.conf` содержит значения по умолчанию для различных параметров командной строки **mke2fs**. Вы можете отредактировать файл, чтобы значения по умолчанию соответствовали вашим потребностям. Например, некоторые утилиты (не в LFS или BLFS) не могут распознать файловую систему `ext4` с включенным параметром `metadata_csum_seed` . **Если** вам нужна такая утилита, вы можете удалить параметр из списка по умолчанию для `ext4` с помощью команды:

```
sed 's/metadata_csum_seed,//' -i /etc/mke2fs.conf
```

Подробности читайте на странице руководства *mke2fs.conf(5)*.

8.81.3. Содержимое пакета E2fsprogs

Установленные программы:	badblocks, chattr, compile_et, debugfs, dumpe2fs, e2freefrag, e2fsck, e2image, e2label, e2mmpstatus, e2scrub, e2scrub_all, e2undo, e4crypt, e4defrag, filefrag, fsck.ext2, fsck.ext3, fsck.ext4, logsave, lsattr, mk_cmds, mke2fs, mkfs.ext2, mkfs.ext3, mkfs.ext4, mklost+found, resize2fs и tune2fs
Установленные библиотеки:	libcom_err.so, libe2p.so, libext2fs.so, и libss.so
Созданные каталоги:	/usr/include/e2p, /usr/include/et, /usr/include/ext2fs, /usr/include/ss, /usr/lib/e2fsprogs, /usr/share/et и /usr/share/ss

Краткое описание

badblocks	Выполняет поиск поврежденных блоков на устройстве (обычно на разделе диска)
chattr	Изменяет атрибуты файлов в файловых системах `ext{234}`
compile_et	Компилятор таблицы ошибок; конвертирует таблицу имен кодов ошибок и сообщений в файл исходного кода на языке C с тем, чтобы ее можно было использовать с библиотекой `com_err`
debugfs	Отладчик файловой системы; его можно использовать для проверки и изменения состояния файловых систем `ext{234}`
dumpe2fs	Выводит информацию о суперблоке и группе блоков для файловой системы, присутствующей на указанном устройстве.
e2freefrag	Сообщает информацию о фрагментации свободного пространства
e2fsck	Используется для проверки и, при необходимости, восстановления файловых систем `ext{234}`
e2image	Используется для сохранения важных данных файловых систем `ext{234}` в файл
e2label	Отображает или изменяет метку файловой системы в файловой системе `ext{234}` на данном устройстве.
e2mmpstatus	Проверяет состояние MMP (Multiple Mount Protection - защита от множественного монтирования) файловой системы `ext4`
e2scrub	Проверяет содержимое смонтированной файловой системы `ext{234}`
e2scrub_all	Проверяет все смонтированные файловые системы `ext{234}` на наличие ошибок
e2undo	Воспроизводит журнал отмены (undo_log) для файловой системы `ext{234}`, обнаруженной на устройстве. [Это можно использовать для отмены неудачной операции программой E2fsprogs.]
e4crypt	Утилита шифрования файловой системы `Ext4`
e4defrag	Онлайн дефрагментатор для файловой системы `ext4`

filefrag	Сообщает о том, насколько сильно может быть фрагментирован конкретный файл
fsck.ext2	По умолчанию проверяет файловые системы `ext2` и является жесткой ссылкой на **e2fsck**
fsck.ext3	По умолчанию проверяет файловые системы `ext3` и является жесткой ссылкой на **e2fsck**
fsck.ext4	По умолчанию проверяет файловые системы `ext4` и является жесткой ссылкой на **e2fsck**
logsave	Сохраняет вывод команды в файл журнала
lsattr	Перечисляет атрибуты файлов во второй расширенной файловой системе.
mk_cmds	Преобразует таблицу имен команд и справочных сообщений в исходный файл C, подходящий для использования с библиотекой подсистемы `libss`
mke2fs	Создает файловую систему `ext{234}` на указанном устройстве
mkfs.ext2	По умолчанию создает файловую систему `ext2` и является жесткой ссылкой на **mke2fs**
mkfs.ext3	По умолчанию создает файловую систему `ext3` и является жесткой ссылкой на **mke2fs**
mkfs.ext4	По умолчанию создает файловую систему `ext4` и является жесткой ссылкой на **mke2fs**
mklost+found	Используется для создания каталога `lost+found` в файловой системе `ext{234}`; предварительно выделяет дисковые блоки для этого каталога, чтобы облегчить задачу **e2fsck**
resize2fs	Может использоваться для увеличения или уменьшения файловой системы `ext{234}`
tune2fs	Позволяет настроить параметры для файловой системы `ext{234}`
`libcom_err`	Стандартная процедура отображения ошибок
`libe2p`	Используется **dumpe2fs**, **chattr**, и **lsattr**
`libext2fs`	Содержит подпрограммы, позволяющие программам пользовательского уровня управлять файловой системой `ext{234}`
`libss`	Используется **debugfs**

8.82. Sysklogd-2.6.1

Пакет Sysklogd содержит программы для логирования системных сообщений, таких как сообщения ядра, когда происходят различные события.

Приблизительное время сборки: менее 0.1 SBU
Требуемое дисковое пространство: 3.9 MB

8.82.1. Установка пакета Sysklogd

Подготовьте пакет к компиляции:

```
./configure --prefix=/usr      \
            --sysconfdir=/etc  \
            --runstatedir=/run \
            --without-logger
```

Скомпилируйте пакет:

```
make
```

Этот пакет не содержит набора тестов.

Установите пакет:

```
make install
```

8.82.2. Настройка Sysklogd

Создайте новый файл `/etc/syslog.conf` , выполнив следующую команду:

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

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

# Do not open any internet ports.
secure_mode 2

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

8.82.3. Содержимое пакета Sysklogd

Установленные программы: syslogd

Краткое описание

syslogd Регистрирует сообщения системных программ для записи в лог [Каждое регистрируемое сообщение содержит как минимум время события, имя хоста, а также имя программы, но это зависит от настроек службы логирования.]

8.83. SysVinit-3.10

Пакет SysVinit содержит программы для управления загрузкой, выполнением и выключением системы.

Приблизительное время сборки:	менее 0.1 SBU
Требуемое дисковое пространство:	2.8 MB

8.83.1. Установка пакета SysVinit

Сначала примените патч, который удаляет несколько программ, установленных другими пакетами и исправляет предупреждение компилятора:

```
patch -Np1 -i ../sysvinit-3.10-consolidated-1.patch
```

Скомпилируйте пакет:

```
make
```

С этим пакетом не поставляется набор тестов.

Установите пакет:

```
make install
```

8.83.2. Содержимое пакета SysVinit

Установленные программы:	bootlogd, fstab-decode, halt, init, killall5, poweroff (ссылка на halt), reboot (ссылка на halt), runlevel, shutdown и telinit (ссылка на init)

Краткое описание

bootlogd	Логирует сообщения при загрузке системы
fstab-decode	Запускает команду с аргументами, заданными в fstab
halt	Обычно вызывает команду **shutdown** с параметром *-h* , за исключением случаев, когда уровень запуска уже равен 0, тогда он посылает ядру сигнал на остановку системы; Кроме этого отмечает в файле `/var/log/wtmp` , что система завершает работу
init	Первый процесс, который запускается после инициализации оборудования; init берет на себя процесс загрузки и запускает все процессы, которые указанны в его файле конфигурации
killall5	Посылает сигнал всем процессам, за исключением процессов в его собственном сеансе; он не завершит свою родительскую оболочку
poweroff	Посылает ядру команду остановить систему и выключить компьютер (смотрите команду **halt**)
reboot	Посылает ядру команду перезагрузить систему (смотрите команду **halt**)
runlevel	Сообщает о предыдущем и текущем уровнях выполнения так, как это указано в последней записи в файле `/run/utmp`
shutdown	Безопасно останавливает систему, отправляя сигналы всем процессам и оповещая всех пользователей, находящихся в системе
telinit	Сообщает **init** на какой уровень выполнения перейти

8.84. Об отладочных символах

Большинство программ и библиотек по умолчанию компилируются с отладочными символами (**gcc** с параметром `-g`). Это означает, что при отладке программы или библиотеки, которые были скомпилированы с использованием отладочной информации, отладчик может предоставить не только адреса памяти, но и имена подпрограмм и переменных.

Включение отладочных символов значительно увеличивает размер программы или библиотеки. Ниже приведена информация по объему пространства, занимаемого отладочными символами:

- Двоичный файл **bash** с отладочными символами: 1200 KB
- Двоичный файл **bash** без отладочных символов: 480 KB (на 60% меньше)
- Файлы Glibc и GCC (`/lib` и `/usr/lib`) с отладочными символами: 87 MB
- Файлы Glibc и GCC без отладочных символов: 16 MB (на 82% меньше)

Размеры могут варьироваться в зависимости от используемого компилятора и библиотеки Си, но программа, в которой были удалены отладочные символы, обычно примерно на 50-80% меньше, чем ее аналог с ними. Поскольку большинство пользователей никогда не будут использовать отладчик в своем программном обеспечении, удаление отладочных символов может освободить много места на диске. В следующем разделе показано, как удалить все отладочные символы из программ и библиотек.

8.85. Удаление отладочных символов

Этот раздел является необязательным. Если предполагаемый пользователь не является программистом и не планирует выполнять какую-либо отладку системного программного обеспечения, размер системы можно уменьшить примерно на 2 ГБ, удалив отладочные символы и некоторые ненужные записи таблицы символов из двоичных файлов и библиотек. Это не вызывает никаких неудобств для обычного пользователя Linux.

Большинство людей, использующих приведенные ниже команды, не испытывают никаких трудностей. Однако легко допустить опечатку и сделать новую систему непригодной для использования. Поэтому перед выполнением команды **strip** рекомендуется сделать резервную копию системы LFS.

Команда **strip** с параметром `--strip-unneeded` удаляет все отладочные символы из двоичного файла или библиотеки. Кроме этого, она удаляет все записи таблицы символов, ненужные компоновщику (для статических библиотек) или динамическому компоновщику (для динамически подключаемых двоичных файлов и общих библиотек).

Отладочные символы из выбранных библиотек сжимаются с помощью Zlib и сохраняются в отдельных файлах. Эта отладочная информация необходима при выполнении регрессионных тестов, с помощью *valgrind* или *gdb* позже, в BLFS.

Обратите внимание, что команда **strip** перезапишет двоичный файл или библиотеку, которую она обрабатывает. Это может привести к сбою процессов, использующих код или данные из файла. Если это затронет сам процесс, выполняющий **strip**, удаляемый двоичный файл или библиотека могут быть уничтожены; это может сделать систему полностью непригодной для использования. Чтобы избежать этого, мы скопируем некоторые библиотеки и двоичные файлы в `/tmp` , очистим их и переустановим с помощью команды **install**. Прочтите статью Раздел 8.2.1, «Проблемы с обновлением», чтобы понять, почему следует использовать команду **install** здесь.

Примечание

Имя загрузчика ELF — ld-linux-x86-64.so.2 в 64-битных системах. и ld-linux.so.2 в 32-битных системах. Конструкция ниже выбирает правильное имя для текущей архитектуры, исключая всё, что заканчивается на `g`, если приведенные ниже команды уже были выполнены.

```
    install -vm755 /tmp/$LIB /usr/lib
    rm /tmp/$LIB
done

cd /usr/lib32
for LIB in $save_usrlib; do
    objcopy --only-keep-debug $LIB $LIB.dbg
    cp $LIB /tmp/$LIB
    strip --strip-unneeded /tmp/$LIB
    objcopy --add-gnu-debuglink=$LIB.dbg /tmp/$LIB
    install -vm755 /tmp/$LIB /usr/lib32
    rm /tmp/$LIB
done


online_usrbin="bash find strip"
online_usrlib="libbfd-2.43.1.so
               libsframe.so.1.0.0
               libhistory.so.8.2
               libncursesw.so.6.5
               libm.so.6
               libreadline.so.8.2
               libz.so.1.3.1
               libzstd.so.1.5.6
               $(cd /usr/lib; find libnss*.so* -type f)"

for BIN in $online_usrbin; do
    cp /usr/bin/$BIN /tmp/$BIN
    strip --strip-unneeded /tmp/$BIN
    install -vm755 /tmp/$BIN /usr/bin
    rm /tmp/$BIN
done

for LIB in $online_usrlib; do
    cp /usr/lib/$LIB /tmp/$LIB
    strip --strip-unneeded /tmp/$LIB
    install -vm755 /tmp/$LIB /usr/lib
    rm /tmp/$LIB
done
for LIB in $online_usrlib; do
    cp /usr/lib32/$LIB /tmp/$LIB
    strip --strip-unneeded /tmp/$LIB
    install -vm755 /tmp/$LIB /usr/lib32
    rm /tmp/$LIB
done

for i in $(find /usr/lib -type f -name \*.so* ! -name \*dbg) \
         $(find /usr/lib -type f -name \*.a)                 \
         $(find /usr/{bin,sbin,libexec} -type f); do
    case "$online_usrbin $online_usrlib $save_usrlib" in
        *$(basename $i)* )
            ;;
        * ) strip --strip-unneeded $i
            ;;
    esac
done
for i in $(find /usr/lib32 -type f -name \*.so* ! -name \*dbg) \
         $(find /usr/lib32 -type f -name \*.a); do
    case "$online_usrbin $online_usrlib $save_usrlib" in
        *$(basename $i)* )
            ;;
        * ) strip --strip-unneeded $i
            ;;
    esac
done

unset BIN LIB save_usrlib online_usrbin online_usrlib
```

Большое количество файлов будет помечено как ошибочные, потому что формат файла не распознан. Эти предупреждения можно смело игнорировать. Они указывают на то, что файлы являются скриптами, а не двоичными файлами.

8.86. Очистка

Наконец, удалите некоторые лишние файлы, оставшиеся после запуска тестов:

```
rm -rf /tmp/{*,.*}
```

Также в каталогах /usr/lib и /usr/libexec есть несколько файлов с расширением .la. Это файлы «архива libtool». Как было сказано ранее, в современной системе Linux файлы .la libtool необходимы только для libltdl. Предполагается, что libltdl не будет загружать библиотеки в LFS, кроме этого известно, что некоторые файлы .la могут нарушить сборку пакетов BLFS. Удалите эти файлы сейчас:

```
find /usr/lib /usr/libexec -name \*.la -delete
find /usr/lib32 -name \*.la -delete
```

Дополнительные сведения об архивных файлах Libtool смотрите в книге BLFS *"О файлах Libtool Archive (.la)"*.

Компилятор, собранный в Глава 6 и Глава 7 все еще установлен, но больше не нужен. Удалите его с помощью команды:

```
find /usr -depth -name $(uname -m)-lfs-linux-gnu\* | xargs rm -rf
```

Наконец, удалите временную учетную запись пользователя 'tester', созданную в начале предыдущей главы.

```
userdel -r tester
```

Глава 9. Системные настройки

9.1. Введение

Загрузка системы Linux включает в себя несколько задач. Процесс должен монтировать как виртуальные, так и реальные файловые системы, инициализировать устройства, активировать файл подкачки, проверять целостность файловых систем, монтировать любые разделы или файл подкачки, устанавливать системные часы, запускать сеть, запускать требуемые системой службы и выполнять любые другие необходимые пользователю задачи. Этот процесс должен быть организован таким образом, чтобы задачи выполнялись в правильном порядке, но в то же время как можно быстрее.

9.1.1. System V

System V это классический процесс загрузки, который использовался в Unix и Unix-подобных системах, таких как Linux с 1983 года. Он состоит из небольшой программы **init**, которая настраивает базовые программы, такие как **login** (через getty) и запускает скрипт. Этот скрипт, обычно именуемый **rc** управляет выполнением дополнительных сценариев, необходимых для инициализации системы.

Программа **init** управляется файлом `/etc/inittab` и разделена по уровням выполнения, которые могут быть выбраны пользователем. В LFS используются следующие уровни:

```
0 — выключение
1 — Однопользовательский режим
2 — Определяемый пользователем
3 — Полный многопользовательский режим
4 — Определяемый пользователем
5 — Полный многопользовательский режим с дисплей-менеджером
6 — перезагрузка
```

Уровень выполнения по умолчанию равен 3 или 5.

Преимущества

- Устоявшаяся, хорошо отлаженная система.
- Легко настраивается.

Недостатки

- Может загружаться медленнее. Загрузка базовой системы LFS в среднем занимает 8-12 секунд, при этом время загрузки измеряется от первого сообщения ядра до запроса на вход в систему. Подключение к сети обычно устанавливается примерно через 2 секунды после запроса на вход в систему.
- Последовательная обработка задач загрузки. Это связано с предыдущим пунктом. Задержка в работе любого процесса, например, проверка файловой системы, приведет к задержке всего процесса загрузки.
- Напрямую не поддерживает дополнительные функции, такие как контрольные группы (cgroups) и диспетчер системных ресурсов для каждого пользователя.
- Добавление init-скриптов требует ручных однотипных действий.

9.2. LFS-Bootscripts-20240825

Пакет LFS-Bootscripts содержит набор скриптов для запуска/остановки системы LFS при загрузке/завершении работы. Файлы конфигурации и процедуры, необходимые для настройки /процесса загрузки, описаны в следующих разделах.

Приблизительное время сборки: менее 0.1 SBU
Требуемое дисковое пространство: 248 KB

9.2.1. Установка пакета LFS-Bootscripts

Установите пакет:

```
make install
```

9.2.2. Содержимое пакета LFS-Bootscripts

Установленные скрипты: checkfs, cleanfs, console, functions, halt, ifdown, ifup, localnet, modules, mountfs, mountvirtfs, network, rc, reboot, sendsignals, setclock, ipv4-static, swap, sysctl, sysklogd, template, udev и udev_retry
Созданные каталоги: /etc/rc.d, /etc/init.d (символическая ссылка), /etc/sysconfig, /lib/services, /lib/lsb (символическая ссылка)

Краткое описание

checkfs Проверяет целостность файловых систем перед их монтированием (за исключением журналируемых и сетевых файловых систем)

cleanfs Удаляет файлы, которые не должны сохраняться между перезагрузками, например, те, которые находятся в директориях `/run/` и `/var/lock/` ; скрипт пересоздает `/run/utmp` и удаляет файлы, которые, возможно, существуют `/etc/nologin` , `/fastboot` , и `/forcefsck`

console Загружает правильную таблицу раскладок для требуемой раскладки клавиатуры; он также устанавливает экранный шрифт

functions Содержит общие функции, такие как проверка ошибок и состояния, которые используются несколькими загрузочными скриптами

halt Останавливает работу системы

ifdown Останавливает сетевое устройство

ifup Инициализирует сетевое устройство

localnet Настраивает имя хоста системы и локальное устройство loopback

modules Загружает модули ядра из списка, который находится в файле `/etc/sysconfig/modules` , используя указанные там же аргументы

mountfs Монтирует все файловые системы, кроме тех, которые помечены как *noauto* или сетевых файловых систем

mountvirtfs Монтирует виртуальные файловые системы, такие как `proc`

network Выполняет настройку сетевых интерфейсов, например, сетевых карт, и настаивает шлюз по-умолчанию (где это применимо)

rc Основной сценарий управления; он отвечает за запуск всех остальных загрузочных скриптов один за другим, в последовательности, определяемой именами символических ссылок на другие скрипты.

reboot	Перезагружает систему
sendsignals	Обеспечивает завершение каждого процесса перед перезагрузкой или остановкой системы.
setclock	Сбрасывает системные часы на местное время, если аппаратные часы не установлены на всемирное координированное время (UTC).
ipv4-static	Предоставляет функциональные возможности, необходимые для назначения статического адреса сетевому интерфейсу.
swap	Включает и отключает файлы и разделы, используемые для подкачки.
sysctl	Загружает в работающее ядро значения системных настроек из файла `/etc/sysctl.conf` , если этот файл существует
sysklogd	Запускает и останавливает демоны журналирования сообщений, выдаваемых системой и ядром.
template	Шаблон для создания пользовательских загрузочных скриптов для других демонов
udev	Подготавливает каталог `/dev` и запускает демон udev
udev_retry	Повторяет неудачные попытки запуска udev uevents и копирует сгенерированные файлы правил из `/run/udev` в `/etc/udev/rules.d` если требуется.

9.3. Взаимодействие с устройствами и модулями

В Глава 8, мы установили демон udev во время сборки udev . Прежде чем мы углубимся в детали того, как работает udev, необходимо кратко рассказать о предыдущих методах взаимодействия с устройствами.

Системы Linux традиционно использовали метод статического создания устройств, при котором огромное количество узлов устройств(иногда буквально тысячи узлов) создавалось в `/dev` , независимо от того, существовали ли соответствующие аппаратные устройства на самом деле. Обычно это делалось с помощью скрипта **MAKEDEV**, который содержал ряд вызовов команды **mknod** с соответствующими основными и второстепенными номерами устройств, для всех возможных вариантов, которые только могут существовать в мире.

Используя метод udev, узлы устройств создаются только для тех устройств, которые обнаружены ядром. Эти узлы устройств создаются каждый раз при загрузке системы; они хранятся в файловой системе `devtmpfs` (виртуальная файловая система, которая полностью находится в оперативной памяти). Узлы не занимают много места в памяти и их общий размер незначителен.

9.3.1. История

В феврале 2000 года, новая файловая система `devfs` была принята в ветку ядра 2.3.46 и была доступна на протяжении выпуска стабильных релизов ветки 2.4. Хотя она и присутствовала в ядре, такой способ динамического создания устройств никогда не получал поддержки от разработчиков ядра.

Основная проблема с подходом, принятым `devfs` была связана с обработкой обнаружения, создания и назначения имен устройствам. Проблема связанная с именованием узлов была самой важной. Общепринято, что если имена устройств можно настраивать, политика именования устройств должна выбираться системными администраторами, а не навязываться разработчиками. Файловая система `devfs` также страдала от состояния гонки, присущего её архитектуре; оно не могло быть исправлено без существенной переработки ядра. `devfs` долгое время была помечена как устаревшая и, наконец, была удалена из ядра в июне 2006 года.

При разработке нестабильной ветки ядра 2.5, позднее, выпущенной как стабильный релиз 2.6, появилась новая виртуальная файловая система `sysfs`. Задача этой файловой системы заключалась в предоставление информации о конфигурации оборудования системы процессам пользовательского пространства. С помощью этого представления, видимого в пользовательском пространстве, стало возможным разработать замену пользовательского пространства для `devfs`.

9.3.2. Реализация Udev

9.3.2.1. Sysfs

Краткое описание файловой системы `sysfs` было представлено выше. Можно задаться вопросом, как `sysfs` получает информацию об устройствах в системе, и о том, какие номера устройств должны использоваться для них. Драйверы, скомпилированные в ядро, регистрируют свои объекты в `sysfs` (внутри `devtmpfs`), по мере обнаружения ядром. Для драйверов, которые скомпилированы в виде модулей, регистрация происходит при его загрузке. После монтирования файловой системы `sysfs` (в каталог `/sys`), данные, зарегистрированные драйверами, в `sysfs`, станут доступны для пользовательского пространства и udevd для обработки (включая модификацию узлов устройств).

9.3.2.2. Создание узла устройства

Файлы устройств создаются ядром в файловой системе `devtmpfs`. Любой драйвер, которому необходимо зарегистрировать узел устройства, будет использовать для этого `devtmpfs` (через системный драйвер ядра). Когда экземпляр `devtmpfs` монтируется в каталог `/dev` , узел устройства будет доступен в пользовательском пространстве с фиксированным именем, разрешениями и владельцем.

Через некоторое время, ядро отправит uevent в **udevd**. На основе правил, которые указанны в файлах в каталогах `/etc/udev/rules.d` , `/lib/udev/rules.d` , и `/run/udev/rules.d` , **udevd** создаст дополнительные символические ссылки на узлы устройств, или сменит разрешения, владельца или группу, или изменит запись (имя) во внутренней базе данных **udevd** для этого объекта.

Правила в этих трёх каталогах пронумерованы и используются совместно. Если **udevd** не может найти правило для устройства, он оставит права доступа и владельца на `devtmpfs`, которые были установлены изначально.

9.3.2.3. Загрузка модуля

Драйверы устройств, скомпилированные в виде модулей ядра могут содержать встроенные псевдонимы. Псевдонимы можно увидеть просмотрев вывод программы **modinfo**, обычно они связаны со специфичными для шины идентификаторами устройств, которые поддерживается модулем. Например, драйвер *snd-fm801* подерживает PCI устройства с идентификатором поставщика 0x1319 и идентификатором устройства 0x0801, и имеет псевдоним `pci:v00001319d00000801sv*sd*bc04sc01i*`. Для большинства устройств, драйвер шины экспортирует псевдонимы драйвера, которые будет обрабатывать устройство через `sysfs`. Например, файл `/sys/bus/pci/devices/0000:00:0d.0/modalias` может содержать строку `pci:v00001319d00000801sv00001319sd00001319bc04sc01i00`. Правила по умолчанию, которые предоставлены Udev, заставят **udevd** вызвать **/sbin/modprobe** с содержимым, которое находится в значении переменной окружения `MODALIAS` uevent (которое должно совпадать с содержимым файла `modalias` в sysfs), тем самым загружая все модули, чьи псевдонимы совпадают в строке после расширения подстановочных знаков

В указанном примере, это означает, что в дополнение к *snd-fm801* будет загружен устаревший (и нежелательный) драйвер *forte*, если он будет доступен. Ниже приведены способы, как можно предотвратить загрузку нежелательных драйверов.

Само ядро также способно загружать модули для сетевых протоколов, файловых систем и поддержки NLS по запросу.

9.3.2.4. Работа с устройствами с горячей заменой или динамическими устройствами

При подключении устройства, например, MP3-плеер, к универсальной последовательной шине (USB), ядро распознает, что устройство подключено, и генерирует событие uevent. Затем это событие обрабатывается **udevd**, как было описано выше.

9.3.3. Проблемы с загрузкой модулей и созданием устройств

Существует несколько возможных проблем, связанных с автоматическим созданием узлов устройств.

9.3.3.1. Модуль ядра не загружается автоматически

Udev загрузит модуль только в том случае, если у него есть псевдоним, специфичный для шины, и драйвер шины правильно экспортирует необходимые псевдонимы в `sysfs`. В других случаях следует организовать загрузку модуля иными способами. Известно, что, начиная с версии Linux-6.10.5, udev, выполняет загрузку правильно написанных драйверов для INPUT, IDE, PCI, USB, SCSI, SERIO, и FireWire устройств.

Чтобы определить, имеет ли требуемый драйвер устройства необходимую поддержку Udev, запустите **modinfo** с именем модуля в качестве аргумента. Далее, попробуйте найти каталог устройства в `/sys/bus` и проверьте, есть ли там файл `modalias`.

Если файл `modalias` существует в `sysfs`, то драйвер, который поддерживает устройство, может обращаться к нему напрямую, но не имеет псевдонима, это ошибка в драйвере. Загрузите драйвер без помощи Udev и ожидайте, что проблема будет исправлена позднее.

Если же в каталоге `/sys/bus` нет файла `modalias`, это означает, что разработчики ядра еще не добавили поддержку `modalias` к этому типу шины. В Linux-6.10.5 это относится к шиной ISA. Ожидайте, что эта проблема будет исправлена в более поздних версиях ядра.

Udev не предназначен для загрузки драйверов «обёрток», таких как *snd-pcm-oss* и не аппаратных драйверов, например, *loop*.

9.3.3.2. Модуль ядра не загружается автоматически и Udev не предназначен для его загрузки

Если модуль «обёртка» только расширяет функциональность, предоставляемую каким-либо другим модулем (например модуль *snd-pcm-oss* расширяет функциональность модуля *snd-pcm*, давая возможность звуковым картам быть доступными для OSS приложений), настройте **modprobe** для загрузки оболочки после того, как Udev загрузит обернутый модуль. Для этого добавьте строку «softdep» в файл, который находится в каталоге `/etc/modprobe.d/<filename>.conf` . Например:

```
softdep snd-pcm post: snd-pcm-oss
```

Обратите внимание, что команда «softdep» разрешает добавлять `pre:` зависимости, или одновременно `pre:` и `post:` зависимости. Обратитесь к документации *modprobe.d(5)* для изучения синтаксиса и возможностей «softdep».

Если рассматриваемый модуль не является обёрткой, и полезен сам по себе, настройте загрузочный скрипт **modules**, чтобы он инициализировался при загрузке системы. Для этого добавьте имя модуля в файл `/etc/sysconfig/modules` в отдельной строке. Этот способ сработает и для модулей-обёрток, но не является оптимальным.

9.3.3.3. Udev загружает какой-то нежелательный модуль

Либо не создавайте модуль, либо занесите его в черный список в файле `/etc/modprobe.d/blacklist.conf` , как это сделано с модулем *forte* в примере ниже:

```
blacklist forte
```

Модули, занесенные в черный список, можно загрузить вручную с помощью явной команды **modprobe**.

9.3.3.4. Udev неправильно создает устройство или делает неправильную символическую ссылку

Это обычно происходит, если правило неожиданно совпадает с другим устройством. Например, плохо написанное поставщиком оборудования правило может соответствовать как диску SCSI(искомое устройство), так и универсальному устройству SCSI (неправильно). Найдите ошибочное правило и исправьте его с помощью команды **udevadm info**.

9.3.3.5. Правило Udev работает ненадежно

Это может быть проявлением предыдущей проблемы. В ином случае, если правило использует атрибуты файловой системы `sysfs`, то это может быть проблемой синхронизации ядра, которая будет исправлена в более поздних версиях ядра. Но вы можете обойти проблему, создав правило, которое ожидает используемый атрибут `sysfs` и добавляет его к файлу правил `/etc/udev/rules.d/10-wait_for_sysfs.rules` (создайте его, если файл не существует). Пожалуйста, сообщите в списке рассылки разработчиков LFS, если это решение вам поможет.

9.3.3.6. Udev не создаёт устройство

Во-первых, убедитесь, что драйвер встроен в ядро или уже загружен как модуль, и, что udev не создает устройство с неправильным именем.

Если драйвер ядра не экспортирует свои данные в `sysfs`, udev не хватает информации, необходимой для создания узла устройства. Это, вероятнее всего, произойдет со сторонними драйверами, которых нет в дереве исходного кода ядра. Создайте статический узел в каталоге `/usr/lib/udev/devices` с

соответствующими старшим/младшим номерами (смотрите файл devices.txt в документации к ядру или документации, предоставленной сторонним поставщиком драйвера). Статический узел будет скопирован в `/dev` с помощью **udev**.

9.3.3.7. Порядок присвоения имен устройствам меняется случайным образом после перезагрузки

Это связано с тем, что udev обрабатывает события uevents и загружает модули параллельно, а значит в непредсказуемом порядке. Это никогда не будет «исправлено». Вы не должны полагаться на то что имена устройств ядра стабильны. Вместо этого создайте свои собственные правила, которые делают символические ссылки со стабильными именами на основе некоторых неизменяемых атрибутов устройства, таких как серийный номер или вывод различных утилит *_id, установленных Udev. Смотрите Раздел 9.4, «Управление устройствами» и Раздел 9.5, «Настройка сети» для примера.

9.3.4. Полезная информация

Дополнительную документацию можно получить на следующих сайтах:

- Реализация пользовательского пространства в `devfs` *http://www.kroah.com/linux/talks/ols_2003_udev_paper/Reprint-Kroah-Hartman-OLS2003.pdf*
- Файловая система `sysfs` *https://www.kernel.org/pub/linux/kernel/people/mochel/doc/papers/ols-2005/mochel.pdf*

9.4. Управление устройствами

9.4.1. Сетевые устройства

Udev по умолчанию присваивает имена сетевым устройствам в соответствии с данными прошивки, BIOS'a или физическими характеристиками, такими как шина, слот или MAC-адрес. Целью такого соглашения об именовании является обеспечение того, чтобы сетевые устройства именовались последовательно, а не основывались на времени обнаружения сетевой карты. Например, в более старых версиях Linux—на компьютере с двумя сетевыми картами производства Intel и Realtek, сетевая карта производства Intel могла стать eth0, а карта Realtek — eth1. Иногда после перезагрузки карты именовались наоборот.

В новой схеме именования, типичными именами сетевых устройств являются enp5s0 или wlp3s0. Если такие имена для вас нежелательны, то может быть реализована традиционная схема именования или своя собственная.

9.4.1.1. Отключение постоянного присвоения имен в параметрах загрузки ядра

Традиционная схема именования - eth0, eth1, и так далее, может быть включена путем добавления параметра **`net.ifnames=0`** в командную строку ядра. Это решение подходит для систем, которые имеют только одно сетевое устройство каждого типа. Часто в ноутбуках несколько сетевых устройств с именами eth0 и wlan0; в таких ноутбуках также может использоваться этот метод. Командная строка указывается в файле конфигурации GRUB. Подробности смотрите на странице Раздел 10.4.4, «Создание файла конфигурации GRUB».

9.4.1.2. Создание пользовательских правил Udev

Схему именования можно настроить, создав пользовательские правила udev. В состав книги включен скрипт, который генерирует начальные правила. Чтобы их сгенерировать, выполните команду:

```
bash /usr/lib/udev/init-net-rules.sh
```

Теперь, проверьте файл `/etc/udev/rules.d/70-persistent-net.rules` , чтобы узнать какое имя с каким сетевым устройством сопоставлено:

```
cat /etc/udev/rules.d/70-persistent-net.rules
```

Примечание

В некоторых случаях, например, когда MAC-адреса были назначены сетевой карте вручную или в виртуальной среде, такой как Qemu или Xen, возможно, файл сетевых правил не будет сгенерирован, поскольку адреса назначаются не последовательно. В таких случаях, этот способ не применим.

Файл начинается с блока комментариев, далее следуют две строки для каждой сетевой карты (NIC). Первая строка представляет собой описание с комментариями и содержит аппаратные идентификаторы (например, поставщика PCI и идентификаторы устройств, если это PCI-карта), а также информацию о драйвере (если его удалось обнаружить). Ни идентификатор оборудования, ни драйвер не используются для определения того, какое имя присвоить интерфейсу; эта информация предназначена только для справки. Вторая строка - это правило udev, которое соответствует этому сетевому адаптеру и фактически присваивает ему имя.

Все правила udev состоят из нескольких ключевых слов, разделенных запятыми и необязательными пробелами. Ниже приведены ключевые слова и пояснения по каждому из них:

- `SUBSYSTEM=="net"` - указывает Udev игнорировать устройства, которые не являются сетевыми картами.
- `ACTION=="add"` - указывает Udev игнорировать правила для событий, отличных от добавления (события "удалить" и "изменить" также происходят, но не требуют переименования сетевых интерфейсов).
- `DRIVERS=="?*"` - существует для того, чтобы Udev проигнорировал подинтерфейсы VLAN или моста (потому что эти подинтерфейсы не имеют драйверов). Эти подинтерфейсы пропускаются, потому что назначенные им имена будут конфликтовать с именами их родительских устройств.
- `ATTR{address}` - значением этого ключевого слова является MAC-адрес сетевой карты.
- `ATTR{type}=="1"` - этот ключ гарантирует выполнение правила соответствующего только основному интерфейсу, при использовании определенных беспроводных драйверов, которые создают несколько виртуальных интерфейсов. Дополнительные интерфейсы пропускаются по той же причине, что и подинтерфейсы VLAN и мост, в ином случае произошел бы конфликт имен.
- `NAME` - значением этого ключевого слова является имя, которое udev присвоит этому интерфейсу.

Значение `NAME` является очень важным. Прежде чем продолжить, убедитесь, что вы знаете, какое имя назначено каждой из сетевых карт и обязательно используйте это значение `NAME` при создании файлов конфигурации сети.

Даже если создан пользовательский файл правил udev, udev все равно может назначить одно или несколько альтернативных имен для сетевой карты на основе физических характеристик. Если пользовательское правило udev переименовывает какой-либо сетевой адаптер, используя имя, уже назначенное в качестве альтернативного имени другого сетевого адаптера, это правило udev не будет выполнено. Если возникает эта проблема, вы можете создать файл конфигурации `/etc/udev/network/99-default.link` с пустой альтернативной политикой назначения, переопределяющей файл конфигурации по умолчанию `/usr/lib/udev/network/99-default.link` :

```
sed -e '/^AlternativeNamesPolicy/s/=.*$/=/'  \
       /usr/lib/udev/network/99-default.link \
     > /etc/udev/network/99-default.link
```

9.4.2. Символические ссылки CD-ROM

Некоторое программное обеспечение, которое вы, возможно, захотите установить позже (например, различные медиаплееры) ожидают, что устройства `/dev/cdrom` или `/dev/dvd` и символические ссылки на CD-ROM или DVD-ROM устройства должны существовать. Кроме того, может быть удобно использовать

эти символические ссылки в `/etc/fstab` . Udev поставляется с файлом сценария, который будет генерировать правила для создания этих символических ссылок, в зависимости от возможностей каждого устройства, но вам нужно решить, какой из двух режимов работы вы хотите использовать.

Во-первых, скрипт может работать в режиме «by-path» (используется по умолчанию для USB и FireWire устройств), где создаваемые им правила зависят от физического пути к CD или DVD устройству. Во-вторых, он может работать в режиме «by-id» (по умолчанию для устройств IDE и SCSI), где создаваемые им правила зависят от строк идентификации, хранящихся в самом устройстве CD или DVD. Путь определяется сценарием Udev **path_id**, а идентификационные строки считываются с оборудования командами **ata_id** или **scsi_id**, в зависимости от того, какой тип устройства у вас есть.

У каждого подхода есть свои преимущества; правильный подход к использованию будет зависеть от того, какие изменения устройств могут произойти. Если вы ожидаете, что физический путь к устройству (порты и/или слоты, в который оно подключено), изменится, например, потому, что вы планируете переместить диск в другой порт IDE или другой разъем USB, то вы должны использовать режим «by-id». С другой стороны, если вы ожидаете, что идентификация устройства изменится, например, потому, что оно может выйти из строя, и вы замените его другим устройством с теми же характеристиками и подключите к тем же разъемам, тогда вы должны использовать режим «by-path».

Если с вашим устройством возможен любой из вариантов, выберите тот, который по вашему мнению случается чаще.

Важно

Внешние устройства (например, привод компакт-дисков, подключенный через USB) не следует подключать методом «by-path», потому что каждый раз, когда устройство подключено в новый внешний порт, изменится его физический путь. Все внешние устройства подвержены этой проблеме, если при написании правил Udev применять режим распознавания по их физическому пути. К тому же, эта проблема не ограничивается CD и DVD-приводами.

Если вы хотите увидеть значения, которые будут использовать скрипты udev, то для требуемого устройства CD-ROM найдите соответствующий каталог в `/sys` (например, это может быть `/sys/block/hdd`) и выполните команду, аналогичную следующей:

```
udevadm test /sys/block/hdd
```

Обратите внимание на строки, содержащие вывод различных идентификаторов *_id. Режим «by-id» будет использовать значение ID_SERIAL если оно существует и не пустое, иначе будет использована комбинация ID_MODEL и ID_REVISION. Режим «by-path» будет использовать значение ID_PATH.

Если режим по умолчанию не подходит для вашей ситуации, то в файл `/etc/udev/rules.d/83-cdrom-symlinks.rules` можно внести следующие изменения (где *mode* является одним из значений «by-id» или «by-path»):

```
sed -e 's/"write_cd_rules"/"write_cd_rules mode"/' \
    -i /etc/udev/rules.d/83-cdrom-symlinks.rules
```

Обратите внимание, что на данный момент, нет необходимости создавать файлы правил или символические ссылки, так как вы смонтировали каталог `/dev` хоста в систему LFS, и мы предполагаем, что символические ссылки уже существуют. Правила и символические ссылки будут создаваться при первой загрузке LFS системы.

Однако, если у вас есть несколько устройств CD-ROM, то символические ссылки, сгенерированные в это время, могут указывать на другие устройства, и иметь различия от хост системы, потому что устройства не будут обнаружены в предсказуемом порядке. Назначения, созданные при первой загрузке системы LFS, будут правильными, проблема возникнет только в том случае, если символические ссылки в обеих

системах указывают на одно и то же устройство. Если потребуется, проверьте (и, возможно, отредактируйте) сгенерированные правила в файле `/etc/udev/rules.d/70-persistent-cd.rules` после загрузки, чтобы убедиться, что назначенные символические ссылки соответствуют тому, что вам нужно.

9.4.3. Работа с дубликатами устройств

Как поясняется в Раздел 9.3, «Взаимодействие с устройствами и модулями», порядок отображения устройства с одинаковой функциональностью в `/dev` является, как правило, случайным. Например, если у вас есть веб камера и TV тюнер, иногда `/dev/video0` ссылается на камеру, а `/dev/video1` ссылается на TV тюнер, а иногда, например, после перезагрузки системы, порядок поменяется на противоположный. Для всех классов оборудования, за исключением звуковых и сетевых карт, это можно исправить, написав правила udev для создания постоянных символических ссылок. Случай с сетевыми картами описан отдельно в Раздел 9.5, «Настройка сети», инструкции по настройке звуковых карт можно найти в *BLFS*.

Для каждого из ваших устройств, которые могут иметь такую проблему (даже если проблема не существует в текущем дистрибутиве Linux), найдите соответствующий каталог в `/sys/class` или `/sys/block` . Для видеоустройств это может быть `/sys/class/video4linux/videoX` . Определите атрибуты, которые однозначно идентифицируют устройство (обычно это идентификаторы поставщика и продукта и/или серийные номера):

```
udevadm info -a -p /sys/class/video4linux/video0
```

Затем напишите правила, которые создают символические ссылки, например:

```
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
```

В результате устройства `/dev/video0` и `/dev/video1` по-прежнему случайным образом ссылаются на TV тюнер и веб-камеру (и, следовательно, никогда не должны использоваться напрямую), но есть символические ссылки /dev/tvtuner и /dev/webcam, которые всегда указывают на правильное устройство.

9.5. Настройка сети

9.5.1. Создание файлов конфигурации сетевого интерфейса

Файлы в каталоге `/etc/sysconfig/` обычно определяют, какие интерфейсы запускаются и выключаются сетевым скриптом. Этот каталог должен содержать файл для каждого настраиваемого интерфейса, например `ifconfig.xyz` , где «xyz» описывает сетевую карту. Имя интерфейса (например, eth0) обычно является подходящим. Каждый файл содержит атрибуты одного интерфейса, такие как его IP-адреса, маски подсетей и так далее. Имя файла должно быть *ifconfig*.

Примечание

Если процедура именования из предыдущего раздела не использовалась, udev будет назначать имена интерфейсов сетевых карт на основе физических характеристик системы, например, enp2s1. Если вы не знаете имя вашего интерфейса, вы всегда можете запустить **ip link** или **ls /sys/class/net** после загрузки системы.

Имена интерфейсов зависят от реализации и конфигурации демона udev, работающего в системе. Демон udev для LFS (установленный в Раздел 8.77, «Udev из Systemd-256.4») не запустится, пока система LFS не будет загружена. Таким образом, имена интерфейсов в системе LFS не всегда можно определить, запустив эти команды в хост-дистрибутиве, *даже в среде chroot*.

Следующая команда создает пример конфигурационного файла для устройства *eth0* со статическим IP-адресом:

```
cd /etc/sysconfig/
cat > ifconfig.eth0 << "EOF"
ONBOOT=yes
IFACE=eth0
SERVICE=ipv4-static
IP=192.168.1.2
GATEWAY=192.168.1.1
PREFIX=24
BROADCAST=192.168.1.255
EOF
```

Значения, выделенные курсивом, должны быть изменены в каждом файле, чтобы правильно настроить интерфейсы.

Если переменной `ONBOOT` присвоено значение `yes`, сценарий загрузки сети System V вызовет сетевую карту (NIC) в процессе загрузки системы. Если задано значение, отличное от `yes` сетевой адаптер будет проигнорирован сетевым скриптом и не будет запущен автоматически. Интерфейсы можно запускать или останавливать вручную с помощью команд **ifup** и **ifdown**.

Переменная `IFACE` определяет имя сетевого интерфейса, например, eth0. Она необходима для всех файлов конфигураций сетевых устройств. Расширение файла должно соответствовать этому значению.

Переменная `SERVICE` определяет метод получения IP-адреса. Пакет LFS-Bootscripts имеет модульный формат назначения IP, а создание дополнительных файлов в каталоге `/lib/services/` позволит использовать другие методы назначения IP. Переменная обычно используется для протокола DHCP (Dynamic Host Configuration Protocol), который рассматривается в книге BLFS

Переменная `GATEWAY` должна содержать IP-адрес шлюза по умолчанию, если таковой имеется. Если нет, то закомментируйте эту строку.

Переменная `PREFIX` указывает количество бит, используемых в подсети. Каждый сегмент IP-адреса состоит из 8 бит. Если маска подсети равна 255.255.255.0, то для указания номера сети используются первые три сегмента (24 бита). Если сетевая маска 255.255.255.240, подсеть использует первые 28 бит. Префиксы длиннее 24 бит обычно используются DSL и кабельными интернет-провайдерами (ISP). В этом примере (PREFIX=24) маска сети — 255.255.255.0. Измените переменную `PREFIX` в соответствии с конфигурацией вашей сети. Если этот параметр не указан, то ПРЕФИКС по умолчанию равен 24.

Для получения дополнительной информации смотрите справочную страницу **ifup**.

9.5.2. Создание файла /etc/resolv.conf

Системе потребуются дополнительные настройки службы доменных имен (DNS) для преобразования имен сети Интернет в IP-адреса и наоборот. Это достигается путем указания IP-адреса DNS-сервера, доступного от провайдера или администратора сети, в `/etc/resolv.conf` . Создайте файл, выполнив следующие действия:

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

domain <Ваше доменное имя>
nameserver <IP-адрес вашего основного DNS-сервера>
nameserver <IP-адрес вашего дополнительного DNS-сервера>

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

Оператор `domain` может быть опущен или заменён оператором `search`. Смотрите справочную страницу resolv.conf для получения подробной информации.

Замените *<IP-адрес вашего основного DNS-сервера>* адресом наиболее подходящего DNS сервера. DNS серверов, может быть указано более одной записи (дополнительные серверы необходимы для возможности резервного переключения). Если вам нужен только один DNS-сервер, удалите вторую строку *nameserver* из файла. DNS-сервер также может выступать шлюзом в локальной сети.

Примечание

Адреса общедоступных DNS серверов Google - 8.8.8.8 и 8.8.4.4.

9.5.3. Настройка имени хоста

В процессе загрузки файл `/etc/hostname` используется для настройки имени хоста системы.

Создайте файл `/etc/hostname` и внесите имя хоста, выполнив команду:

```
echo "<lfs>" > /etc/hostname
```

<lfs> замените на имя вашего компьютера. Не вносите сюда полное доменное имя(FQDN). Эта информация помещается в файл `/etc/hosts` .

9.5.4. Настройка файла /etc/hosts

Выберите полное доменное имя (FQDN) и возможные псевдонимы для использования в файле `/etc/hosts` . Если вы используете статические IP-адреса, вам также необходимо определиться с IP-адресом. Синтаксис для записи в файле hosts следующий:

```
IP_address myhost.example.org aliases
```

Если компьютер не должен быть виден в Интернете (т. е. нет зарегистрированного домена и действительного блока назначенных IP-адресов — у большинства пользователей этого нет), убедитесь, что IP-адрес находится в диапазоне внутренних сетевых IP-адресов. Допустимые диапазоны:

```
Диапазон адресов локальной сети      Стандартный префикс
10.0.0.1 - 10.255.255.254           8
172.x.0.1 - 172.x.255.254           16
192.168.y.1 - 192.168.y.254         24
```

x может быть любым числом в диапазоне 16-31. y может быть любым числом в диапазоне 0-255.

Правильным IP адресом в локальной сети может быть 192.168.1.1.

Если компьютер должен быть виден в Интернете, действительным полным доменным именем может быть само имя домена или строка, полученная путем объединения префикса (часто имени хоста) и домена с помощью символа «.». Кроме того, вам необходимо обратиться к администратору домена, чтобы связать полное доменное имя и ваш общедоступный IP-адрес.

Даже если компьютер не виден из Интернета, полное доменное имя все равно необходимо для правильной работы некоторых программ, таких как MTA. Для этих целей можно использовать специальное полное доменное имя `localhost.localdomain` .

Создайте файл `/etc/hosts` , выполнив команду:

```
cat > /etc/hosts << "EOF"
# Begin /etc/hosts

127.0.0.1 localhost.localdomain localhost
127.0.1.1 <FQDN> <HOSTNAME>
<192.168.1.1> <FQDN> <HOSTNAME> [alias1] [alias2 ...]
::1       localhost ip6-localhost ip6-loopback
ff02::1   ip6-allnodes
ff02::2   ip6-allrouters

# End /etc/hosts
EOF
```

Значения *<192.168.1.1>* , *<FQDN>*, и *<HOSTNAME>* должны быть изменены в соответствии с предпочтениями пользователя или требованиями сети (если имеется IP-адрес выданный сетевым/системным администратором и машина подключена к существующей сети). Необязательные параметры могут быть опущены.

9.6. Настройка и использование загрузочных скриптов System V

9.6.1. Как работают загрузочные скрипты System V?

В этой версии LFS используется специальное средство загрузки под названием SysVinit, основанное на наборе уровней выполнения (*run-levels*). Процедура загрузки может сильно отличаться от одной системы к другой; и тот факт, что в конкретном дистрибутиве Linux все работало должным образом, не гарантирует, что оно будут работать так же в LFS. LFS работает по-своему, но соблюдает общепринятые стандарты.

Существует альтернативная система загрузки, которая называется **systemd**. Мы не будем обсуждать этот вариант загрузки здесь. Для получения подробного описания, посетите *https://www.linux.com/training-tutorials/understanding-and-using-systemd/*.

SysVinit (в дальнейшем именуемый «init») использует схему уровней выполнения. Существует семь уровней выполнения, пронумерованных от 0 до 6. (На самом деле уровней больше, но они предназначены для особых случаев и обычно не используются. Подробности смотрите в руководстве *init(8)*). Каждый из них соответствует действиям, которые компьютер должен выполнить при запуске и выключении. Уровень выполнения по умолчанию — 3. Ниже приведено описание различных уровней в том виде, в каком они реализованы в LFS:

0: выключение компьютера
1: однопользовательский режим
2: зарезервировано для настройки, в остальном аналогично 3
3: многопользовательский режим с поддержкой сети
4: зарезервировано для настройки, в остальном аналогично 3
5: то же, что и 4, обычно используется для входа в систему с графическим интерфейсом (например, **gdm** от GNOME или **lxdm** от LXDE)
6: перезагрузка компьютера

Примечание

Раньше, много лет назад, уровень выполнения 2 обозначался как «многопользовательский режим без поддержки сети», тогда, несколько пользователей могли подключаться к системе через последовательные порты. В сегодняшних условиях это не имеет смысла, и мы обозначаем этот уровень как «зарезервировано».

9.6.2. Настройка SysVinit

Во время инициализации ядра первой запускаемой программой (если она не переопределена в командной строке) является **init**. Эта программа считывает файл инициализации `/etc/inittab`. Создайте этот файл:

```
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:S06:once:/sbin/sulogin
s1:1:respawn:/sbin/sulogin

1:2345:respawn:/sbin/agetty --noclear tty1 9600
2:2345:respawn:/sbin/agetty tty2 9600
3:2345:respawn:/sbin/agetty tty3 9600
4:2345:respawn:/sbin/agetty tty4 9600
5:2345:respawn:/sbin/agetty tty5 9600
6:2345:respawn:/sbin/agetty tty6 9600

# End /etc/inittab
EOF
```

Пояснения по содержимому этого файла инициализации находится на справочной странице *inittab*. Для LFS основной командой является **rc**. В приведенном выше файле инициализации, **rc** будет выполнять все сценарии, начинающиеся с символа S в каталоге `/etc/rc.d/rcS.d`, за которыми следуют все скрипты, начинающиеся с S в каталоге `/etc/rc.d/rc?.d`, где знак вопроса задается значением initdefault.

Для удобства, скрипт **rc** считывает библиотеку функций из `/lib/lsb/init-functions`. Эта библиотека также читает необязательный файл конфигурации `/etc/sysconfig/rc.site`. В этот файл можно поместить любой из параметров конфигурации системы, описанных в последующих разделах, что позволяет объединить все параметры системы в одном файле.

Для удобства отладки скрипт functions также записывает весь вывод в `/run/var/bootlog`. Поскольку каталог `/run` является tmpfs, этот файл не сохраняется при загрузке; однако его содержимое добавляется в файл `/var/log/boot.log` по окончании процесса загрузки.

9.6.2.1. Изменение уровней выполнения

Изменить уровень выполнения можно с помощью команды **init** ***<runlevel>***, где *<runlevel>* - это целевой уровнем выполнения. Например, чтобы перезагрузить компьютер, пользователь должен выполнить команду **init 6**, которая является псевдонимом для команды **reboot**. Аналогично, init 0 является псевдонимом для команды **halt**.

В каталоге `/etc/rc.d` есть ещё несколько каталогов, которые выглядят как `rc?.d` (где ? - номер уровня выполнения), все они содержат ряд символических ссылок. Некоторые ссылки начинаются с *K*, другие начинаются с *S*, и все они содержат две цифры после начальной буквы. K означает остановить (убить) службу, а S означает запустить службу. Числа определяют порядок выполнения сценариев от 00 до 99—чем

меньше число, тем раньше запускается скрипт. Когда **init** переключается на другой уровень выполнения, соответствующие службы либо запускаются, либо останавливаются, в зависимости от выбранного уровня выполнения.

Реальные скрипты находятся в каталоге `/etc/rc.d/init.d` . Они выполняют фактическую работу, и символические ссылки указывают на них. Ссылки K и S указывают на один и тот же скрипт в `/etc/rc.d/init.d` . Это связано с тем, что скрипты могут вызываться с разными параметрами, такими как *`start`*, *`stop`*, *`restart`*, *`reload`*, и *`status`*. Когда встречается ссылка K, соответствующий скрипт запускается с аргументом *`stop`*. Когда встречается S-ссылка, соответствующий скрипт запускается с аргументом *`start`*.

Ниже приведено описание к аргументам скриптов:

`start`
: Служба запущена.

`stop`
: Служба остановлена.

`restart`
: Служба остановлена и снова запущена.

`reload`
: Конфигурация сервиса обновлена. Используется после изменения файла конфигурации службы, когда перезапуск не требуется.

`status`
: Сообщает, запущена ли служба и с какими ИД процессов.

Не стесняйтесь изменять работу процесса загрузки (в конце концов, это ваша собственная система LFS). Приведенные здесь файлы являются примером того, как это можно сделать.

9.6.3. Загрузочные скрипты Udev

Инит-скрипт `/etc/rc.d/init.d/udev` запускает **udevd**, который инициализирует все устройства "холодного подключения", которые уже были созданы ядром, и ожидает выполнения всех правил. Скрипт также отключает обработчик uevent по умолчанию `/sbin/hotplug` . Это сделано потому, что ядру больше не нужно обращаться к внешнему бинарному файлу. Вместо этого **udevd** будет прослушивать сокет netlink на предмет событий, которые вызывает ядро.

Инит-скрипт **/etc/rc.d/init.d/udev_retry** решает задачу повторного запуска событий для подсистем, правила которых могут зависеть от файловых систем, которые не монтируются до запуска скрипта **mountfs** (в частности, его вызывают `/usr` и `/var`). Этот скрипт запускается после скрипта **mountfs**, поэтому эти правила (при повторном запуске) должны сработать во второй раз. Он настраивается в файле `/etc/sysconfig/udev_retry` ; любые слова в этом файле, кроме комментариев, являются именами подсистем, которые запускаются при повторной попытке. Чтобы найти подсистему устройства, используйте **udevadm info --attribute-walk <device>**, где <device> - это абсолютный путь в /dev или /sys, такой как /dev/sr0 или /sys/class/rtc.

Для получения информации о загрузке модуля ядра и udev смотрите Раздел 9.3.2.3, «Загрузка модуля».

9.6.4. Настройка системного времени

Скрипт **setclock** считывает время с аппаратных часов, также известных как часы BIOS или CMOS (Complementary Metal Oxide Semiconductor). Если на аппаратных часах установлено время UTC (всемирное скоординированное время), этот скрипт преобразует полученное значение в локальное время, используя

файл `/etc/localtime` (который сообщает программе **hwclock**, в какой временной зоне находится пользователь). Невозможно определить, установленное на аппаратных часах время UTC или же локальное, поэтому необходимо указать его вручную.

Скрипт **setclock** запускается системой udev при загрузке, когда ядро определяет возможности аппаратного обеспечения. Также его можно запустить вручную с параметром stop, чтобы сохранить системное время в аппаратные часы CMOS.

Если Вы не можете вспомнить, установлено на аппаратных часах UTC или локальное время, запустите команду **`hwclock --localtime --show`**. Она отобразит текущее время в соответствии с аппаратными часами. Если оно совпадает с тем, что показывают ваши настенные/наручные часы, значит на часах CMOS установлено локальное время. Если вывод **hwclock** не совпадает с локальным временем, скорее всего, это UTC. Проверьте это, добавив или вычитая нужное количество часов для вашего часового пояса ко времени, отображаемому **hwclock**. Например, если ваш часовой пояс это MSK, также известный как GMT +0300, то нужно вычесть три часа из локального времени.

Измените значение переменной `UTC` ниже на *`0`* (ноль), если на аппаратных часах установлено *HE* UTC.

Создайте новый файл `/etc/sysconfig/clock` выполнив:

```
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
```

Хороший совет, объясняющий, как обращаться с временем в LFS, доступен по адресу *https://mirror.linuxfromscratch.ru/hints/downloads/files/time.txt*. Там объясняются такие вопросы, как часовые пояса, UTC и переменная окружения `TZ`.

Примечание

Параметры CLOCKPARAMS и UTC также могут быть указаны в файле `/etc/sysconfig/rc.site` .

9.6.5. Настройка консоли Linux

Эта секция описывает настройку скрипта **console**, который устанавливает раскладку клавиатуры, шрифт консоли и уровень подробности информации, выводимой ядром на консоль. Если Вы не планируете использовать символы, не соответствующие стандарту ASCII (например, знак копирайта, символы фунта и евро), и собираетесь печатать только в английской раскладке, то можете пропустить большую часть секции. Без файла конфигурации (или эквивалентных настроек в `rc.site`), скрипт **console** не будет ничего делать.

Скрипт **console** считывает конфигурацию из файла `/etc/sysconfig/console` . Решите для себя, какую раскладку клавиатуры и какой шрифт намерены использовать. В этом также могут помочь различные HOWTO для конкретных языков; смотрите *https://tldp.org/HOWTO/HOWTO-INDEX/other-lang.html*. Если вы все еще сомневаетесь, посмотрите в каталогах `/usr/share/keymaps` и `/usr/share/consolefonts` допустимые раскладки и экранные шрифты. Прочтите справочные страницы *loadkeys(1)* и *setfont(8)*, чтобы определить правильные аргументы для этих программ.

Файл `/etc/sysconfig/console` должен содержать строки вида: `ПЕРЕМЕННАЯ=значение`. Допустимы следующие переменные:

LOGLEVEL

Эта переменная задает уровень подробности сообщений, посылаемых ядром на системную консоль. Значение этой переменной передается в качестве аргумента утилите **dmesg -n**. Допустимы уровни от 1(нет сообщений) до 8. По умолчанию 7, что является довольно подробным.

KEYMAP

Эта переменная указывает аргументы для программы **loadkeys**, обычно это имя загружаемой раскладки, например, it. Если эта переменная не установлена, загрузочный скрипт не будет запускать программу loadkeys, и будет использоваться раскладка по умолчанию. Обратите внимание, что некоторые раскладки имеют несколько версий с одним и тем же именем (cz и его варианты в qwerty/ и qwertz/, es в olpc/ и qwerty/ и trf в fgGIod/ и qwerty/). В этих случаях также следует указать родительский каталог (например, qwerty/es), чтобы обеспечить загрузку правильной раскладки.

KEYMAP_CORRECTIONS

Эта (крайне редко используемая) переменная задает аргументы для второго вызова программы **loadkeys**. Она полезна, если стандартная раскладка вас не совсем устраивает и необходимо внести небольшую корректировку. Например, чтобы добавить символ евро в раскладку, которая его не содержит, присвойте этой переменной значение euro2.

FONT

Эта переменная определяет аргументы для программы **setfont**. Обычно она включает в себя имя шрифта -m и имя загружаемой таблицы символов. Например, чтобы загрузить шрифт «lat1-16» вместе с таблицей символов «8859-1» (подходит для США), установите эту переменную в lat1-16 -m 8859-1 . В режиме UTF-8 ядро использует таблицу символов для преобразования 8-битных кодов клавиш в UTF-8, поэтому аргумент параметра "-m" должен быть установлен на кодировку соответствующих кодов клавиш в раскладке.

UNICODE

Присвойте этой переменной значение 1, yes или true, чтобы переключить консоль в режим UTF-8. Это полезно при использовании локали, основанной на UTF-8, и не рекомендуется в иных случаях.

LEGACY_CHARSET

Для многих раскладок клавиатуры в пакете Kbd не существует готового Unicode-варианта. Скрипт **console** будет на лету конвертировать имеющуюся раскладку в UTF-8, если присвоить этой переменной имя доступной не-UTF-8 раскладки.

Несколько примеров:

- Мы будем использовать C.UTF-8 в качестве локали для интерактивных сеансов в консоли Linux в Раздел 9.7, «Настройка системной локали,», поэтому нам следует установить значение переменной UNICODE в 1. Консольные шрифты, поставляемые пакетом Kbd, содержащие глифы для всех символов сообщений программы в C.UTF-8 являются LatArCyrHeb*.psfu.gz , LatGrkCyr*.psfu.gz , Lat2-Terminus16.psfu.gz и pancyrillic.f16.psfu.gz из каталога /usr/share/consolefonts (в других поставляемых консольных шрифтах отсутствуют глифы некоторых символов Unicode, таких как левые/правые кавычки и английское тире). Поэтому установите один из них, например Lat2-Terminus16.psfu.gz , в качестве шрифта консоли по умолчанию:

```
cat > /etc/sysconfig/console << "EOF"
# Begin /etc/sysconfig/console

UNICODE="1"
FONT="Lat2-Terminus16"

# End /etc/sysconfig/console
EOF
```

- Для русскоязычных пользователей может подойти такой вариант:

```
cat > /etc/sysconfig/console << "EOF"
# Begin /etc/sysconfig/console

UNICODE="1"
KEYMAP="ruwin_alt_sh-UTF-8"
FONT="cyr-sun16"

# End /etc/sysconfig/console
EOF
```

Примечание

Для поиска нужной раскладки, выполните команду: **find /usr/share/keymaps -type f**

`ruwin_alt-UTF-8` - переключение раскладки по Alt.

`ruwin_alt_sh-UTF-8` - переключение раскладки по Alt+Shift.

`ruwin_cplk-UTF-8` - переключение раскладки по Caps Lock.

`ruwin_ct_sh-UTF-8` - переключение раскладки по Ctrl+Shift.

`ruwin_ctrl-UTF-8` - переключение раскладки по Ctrl.

Посмотреть список доступных шрифтов можно с помощью следующей команды: **ls /usr/share/consolefonts**

- Для не-Unicode настройки необходимы только переменные KEYMAP и FONT. Например, для польских пользователей может подойти такой вариант:

```
cat > /etc/sysconfig/console << "EOF"
# Begin /etc/sysconfig/console

KEYMAP="pl2"
FONT="lat2a-16 -m 8859-2"

# End /etc/sysconfig/console
EOF
```

- Как упоминалось выше, иногда бывает необходимо подкорректировать раскладку. Следующий пример добавляет символ евро к немецкой раскладке:

```
cat > /etc/sysconfig/console << "EOF"
# Begin /etc/sysconfig/console

KEYMAP="de-latin1"
KEYMAP_CORRECTIONS="euro2"
FONT="lat0-16 -m 8859-15"
UNICODE="1"

# End /etc/sysconfig/console
EOF
```

- Ниже приведен пример с поддержкой Unicode для болгарского языка, где существует стандартная раскладка 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
```

- Из-за использования 512-символьного шрифта LatArCyrHeb-16 в предыдущем примере, яркие цвета больше не доступны в консоли Linux, если используется фреймбуфер. Если Вы хотите использовать яркие цвета без фреймбуфера и можете обходиться без символов, не относящихся к вашему языку, тогда можно использовать специфичный для вашего языка 256-символьный шрифт, как показано ниже:

```
cat > /etc/sysconfig/console << "EOF"
# Begin /etc/sysconfig/console

UNICODE="1"
KEYMAP="bg_bds-utf8"
FONT="cyr-sun16"

# End /etc/sysconfig/console
EOF
```

- Следующий пример демонстрирует автоматическое преобразование раскладки из ISO-8859-15 в UTF-8 и включает "мертвые" клавиши в режиме 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
```

- Некоторые раскладки включают в себя "мертвые" клавиши (то есть клавиши, нажатие которых само по себе не приводит к появлению на экране символа, но которые влияют на символ, генерируемый следующей клавишей) или определяют слияние символов (например: «нажмите Ctrl+. A E чтобы получить Æ» в раскладке по умолчанию). Linux-6.10.5 правильно интерпретирует "мертвые" клавиши и слияния, только когда исходные символы имеют 8-битные коды. Эта особенность не влияет на раскладки для европейских языков, поскольку в них "сливаются" два ASCII-символа или добавляются подчеркивания к неподчеркнутым ASCII-символам. Однако, в режиме UTF-8 могут быть проблемы, например, для греческого языка, когда необходимо подчеркнуть символ α. Решением в этой ситуации будет отказ от использования UTF-8 или установка графической системы X Window, не имеющих подобных ограничений.

- Для китайского, японского, корейского и некоторых других языков невозможно настроить консоль Linux так, чтобы она отображала все необходимые символы. Пользователи, которым требуются эти языки, должны установить систему X Window, шрифты, покрывающие необходимый диапазон символов, и правильный метод ввода (например, SCIM поддерживает большое число разнообразных языков).

Примечание

Файл /etc/sysconfig/console управляет только локализацией текстовой консоли Linux. Он никак не влияет на настройки раскладки клавиатуры и шрифтов в системе X Window, в сессиях SSH или на последовательном терминале. В этих ситуациях ограничения, описанные в двух расположенных выше абзацах, не применяются.

9.6.6. Создание файлов при загрузке

Иногда необходимо создавать файлы во время загрузки. Например, часто требуется каталог /tmp/.ICE-unix . Это можно сделать, создав запись в скрипте /etc/sysconfig/createfiles . Формат этого файла описан в комментариях файла конфигурации по умолчанию.

9.6.7. Настройка скрипта Sysklogd

Скрипт sysklogd вызывает программу **syslogd** как часть инициализации System V. Параметр *-m 0* отключает периодическую (по умолчанию - каждые 20 минут) запись временных меток в файлы журналов, производимую **syslogd**. Если Вам необходимо включить периодическую запись временных меток, отредактируйте файл /etc/sysconfig/rc.site и присвойте переменной SYSKLOGD_PARMS требуемое значение. Например, чтобы сбросить все параметры, присвойте переменной пустое значение:

```
SYSKLOGD_PARMS=
```

Дополнительные параметры смотрите в **man syslogd**.

9.6.8. Файл rc.site

Необязательный файл /etc/sysconfig/rc.site содержит настройки, автоматически применяемые всеми загрузочными скриптами. Этот файл может содержать парамеры, обычно указываемые в файлах hostname, console и clock из каталога /etc/sysconfig/ . Если значение одной и той же переменной присваивается, как в одном из этих файлов, так и в rc.site , приоритет имеет значение из специализированного файла.

rc.site также содержит параметры, которые могут настраивать другие аспекты процесса загрузки. Установка переменной IPROMPT позволит выборочно запускать загрузочные скрипты. Другие параметры описаны в комментариях к файлу. Версия файла по умолчанию выглядит следующим образом:

```
# rc.site
# Optional parameters for boot scripts.

# Distro Information
# These values, if specified here, override the defaults
#DISTRO="Linux From Scratch" # The distro name
#DISTRO_CONTACT="lfs-dev@lists.linuxfromscratch.org" # Bug report address
#DISTRO_MINI="LFS" # Short name used in filenames for distro config

# Define custom colors used in messages printed to the screen

# Please consult `man console_codes` for more information
# under the "ECMA-48 Set Graphics Rendition" section
#
# Warning: when switching from a 8bit to a 9bit font,
# the linux console will reinterpret the bold (1;) to
# the top 256 glyphs of the 9bit font.  This does
# not affect framebuffer consoles

# These values, if specified here, override the defaults
#BRACKET="\\033[1;34m" # Blue
#FAILURE="\\033[1;31m" # Red
#INFO="\\033[1;36m"    # Cyan
#NORMAL="\\033[0;39m"  # Grey
#SUCCESS="\\033[1;32m" # Green
```

```
#WARNING="\\033[1;33m" # Yellow

# Use a colored prefix
# These values, if specified here, override the defaults
#BMPREFIX="      "
#SUCCESS_PREFIX="${SUCCESS}  *  ${NORMAL} "
#FAILURE_PREFIX="${FAILURE}*****${NORMAL} "
#WARNING_PREFIX="${WARNING} *** ${NORMAL} "

# Manually set the right edge of message output (characters)
# Useful when resetting console font during boot to override
# automatic screen width detection
#COLUMNS=120

# Interactive startup
#IPROMPT="yes" # Whether to display the interactive boot prompt
#itime="3"    # The amount of time (in seconds) to display the prompt

# The total length of the distro welcome string, without escape codes
#wlen=$(echo "Welcome to ${DISTRO}" | wc -c )
#welcome_message="Welcome to ${INFO}${DISTRO}${NORMAL}"

# The total length of the interactive string, without escape codes
#ilen=$(echo "Press 'I' to enter interactive startup" | wc -c )
#i_message="Press '${FAILURE}I${NORMAL}' to enter interactive startup"

# Set scripts to skip the file system check on reboot
#FASTBOOT=yes

# Skip reading from the console
#HEADLESS=yes

# Write out fsck progress if yes
#VERBOSE_FSCK=no

# Speed up boot without waiting for settle in udev
#OMIT_UDEV_SETTLE=y

# Speed up boot without waiting for settle in udev_retry
#OMIT_UDEV_RETRY_SETTLE=yes

# Skip cleaning /tmp if yes
#SKIPTMPCLEAN=no

# For setclock
#UTC=1
#CLOCKPARAMS=

# For consolelog (Note that the default, 7=debug, is noisy)
#LOGLEVEL=7

# For network
#HOSTNAME=mylfs

# Delay between TERM and KILL signals at shutdown
#KILLDELAY=3

# Optional sysklogd parameters
#SYSKLOGD_PARMS="-m 0"

# Console parameters
#UNICODE=1
#KEYMAP="de-latin1"
#KEYMAP_CORRECTIONS="euro2"
#FONT="lat0-16 -m 8859-15"
#LEGACY_CHARSET=
```

9.6.8.1. Настройка скриптов загрузки и завершения работы

Загрузочные скрипты LFS загружают и завершают работу системы довольно эффективно, но есть несколько настроек, которые вы можете внести в файл rc.site, чтобы еще больше повысить скорость и настроить сообщения в соответствии с вашими предпочтениями. Чтобы сделать это, измените настройки в приведенном выше файле `/etc/sysconfig/rc.site` .

- Во время работы загрузочного скрипта `udev` происходит вызов **udev settle**, для завершения которого требуется некоторое время. Это время может и не потребоваться в зависимости от конфигурации устройств в системе. Если у вас имеются только простые разделы и одна сетевая карта, процессу загрузки, вероятно, не нужно будет ждать завершения работы этой команды. Чтобы пропустить её, установите переменную OMIT_UDEV_SETTLE=y.
- Скрипт загрузки `udev_retry` также по умолчанию запускает **udev settle**. Команда необходима только тогда, когда каталог `/var` смонтирован в отдельный раздел. Это связано с тем, что часам нужен доступ к файлу `/var/lib/hwclock/adjtime` . Для других настроек также может потребоваться дождаться завершения udev, но в большинстве случаев в этом нет необходимости. Пропустите команду, установив переменную OMIT_UDEV_RETRY_SETTLE=y.
- По умолчанию проверка файловой системы выполняется в "тихом" режиме. Это может показаться задержкой во время процесса загрузки. Чтобы включить вывод **fsck**, установите переменную VERBOSE_FSCK=y.
- При перезагрузке вы, возможно, захотите полностью пропустить проверку файловой системы, **fsck**. Чтобы сделать это, либо создайте файл `/fastboot` , либо перезагрузите систему командой **/sbin/shutdown -f -r now**. С другой стороны, вы можете принудительно проверить все файловые системы, создав `/forcefsck` или запустив **shutdown** с параметром `-F` вместо `-f` .

  Установка переменной FASTBOOT=y отключит **fsck** во время процесса загрузки до тех пор, пока она не будет удалена. Это не рекомендуется делать на постоянной основе.
- Обычно все файлы в каталоге `/tmp` удаляются во время загрузки. В зависимости от количества имеющихся файлов или каталогов это может привести к заметной задержке в процессе загрузки. Чтобы пропустить удаление этих файлов, установите переменную SKIPTMPCLEAN=y.
- Во время завершения работы, **init** отправляет сигнал TERM каждой запущенной программе (например, agetty), ожидает установленное время (по умолчанию 3 секунды), затем посылает каждому процессу сигнал завершения(KILL) и снова ждёт. Этот процесс повторяется в сценарии **sendsignals** для любых процессов, которые не завершаются их собственными скриптами. Задержка для **init** может быть установлена путем передачи параметра. Например, чтобы устранить задержку в **init**, передайте параметр -t0 при выключении или перезагрузке (например, **/sbin/shutdown -t0 -r now**). Задержку для скрипта **sendsignals** можно пропустить, установив параметр KILLDELAY=0.

9.7. Настройка системной локали

Некоторые переменные окружения, необходимые для поддержки вашего языка. Правильная их установка влияет на:

- Выходные данные программ, переводятся на ваш родной язык
- Правильную интерпретацию символов в буквы, цифры и другие классы. Это необходимо для того, чтобы **bash** правильно принимал не-ASCII символы, в командной строке в неанглоязычных языковых системах
- Правильную для страны сортировку по алфавиту
- Подходящий формат бумаги по умолчанию

- Правильное форматирование денежных значений, значений времени и дат

Ниже замените *<ll>* двухбуквенным кодом нужного вам языка (например en), а *<CC>* двухбуквенным кодом соответствующей страны (например GB). *<charmap>* нужно заменить на каноническую кодировку для выбраной вами локали. Также могут присутствовать необязательные модификаторы, такие как @euro.

Список всех локалей, поддерживаемых Glibc, можно получить, выполнив следующую команду:

```
locale -a
```

Таблицы символов могут иметь несколько синонимов. Например ISO-8859-1 так же называют iso8859-1 и iso88591. Некоторые приложения не могут корректно обрабатывать различные синонимы (например UTF-8 должно быть указано как UTF-8 , а не utf8), поэтому в большинстве случаев безопаснее всего выбрать каноническое имя для конкретной локали. Для определения канонического имени локали выполните команду ниже, заменив *<locale name>* на вывод **locale -a** для желаемой локали (например en_GB.iso88591 в нашем примере).

```
LC_ALL=<locale name> locale charmap
```

Для локали en_GB.iso88591 приведенная выше команда напечатает:

```
ISO-8859-1
```

Окончательная настройка локали будет выглядеть так: en_GB.ISO-8859-1 . Важно, чтобы локаль, найденная с помощью приведенной выше методики, была проверена перед её добавлением в файлы запуска Bash:

```
LC_ALL=<locale name> locale language
LC_ALL=<locale name> locale charmap
LC_ALL=<locale name> locale int_curr_symbol
LC_ALL=<locale name> locale int_prefix
```

Приведенные выше команды должны вывести название языка, кодировку символов, используемую в локали, местную валюту и телефонный код страны. Если какая-либо из команд завершается с сообщением об ошибке, похожим на указанное ниже, это означает, что ваша локаль либо не была установлена в Главе 8, либо не поддерживается стандартной установкой Glibc.

```
locale: Cannot set LC_* to default locale: No such file or directory
```

Если это произойдет, вам следует либо установить желаемую локаль с помощью команды **localedef**, либо рассмотреть возможность выбора другой локали. Дальнейшие инструкции не предполагают таких сообщений об ошибках от Glibc.

Другие пакеты также могут работать некорректно (но не всегда будут отображать какие-либо сообщения об ошибках), если название локали не соответствует их ожиданиям. В таких случаях изучите, какие ещё дистрибутивы Linux поддерживают ваш язык, возможно, это даст некоторую полезную информацию

Программа оболочки **/bin/bash** (далее именуемая как «оболочка») использует начальный набор файлов, чтобы помочь создать среду для запуска. Каждый файл имеет определенное назначение и может по-разному влиять на вход в систему и интерактивное окружение. Файлы в каталоге /etc содержат глобальные настройки. Если в домашнем каталоге существуют эквивалентные файлы, они могут переопределить глобальные настройки.

Интерактивная оболочка входа запускается после успешного входа с использованием **/bin/login**, прочитав файл /etc/passwd . Интерактивная оболочка без входа в систему запускается из командной строки (например, [prompt]$/**bin/bash**). Неинтерактивная оболочка обычно присутствует когда выполняется скрипт оболочки. Он неинтерактивен, поскольку обрабатывает скрипт и не ожидает ввода пользователем между командами.

Создайте файл `/etc/profile` после того, как будут определены правильные настройки локали, чтобы установить желаемый язык, но вместо этого установите локаль `C.UTF-8` при запуске консоли Linux (чтобы программы не выводили символы, которые консоль Linux не может отобразить):

```
cat > /etc/profile << "EOF"
# Begin /etc/profile

for i in $(locale); do
  unset ${i%=*}
done

if [[ "$TERM" = linux ]]; then
  export LANG=C.UTF-8
else
  export LANG=<ll>_<CC>.<charmap><@modifiers>
fi

# End /etc/profile
EOF
```

Локаль `C` (используемая по умолчанию) и `en_US` (одна из рекомендуемых для англоязычных пользователей в Соединенных Штатах) это разные локали. `C` использует 7-битный набор символов US-ASCII и обрабатывает байты с установленным старшим битом как недопустимые символы. Вот почему, например, команда **ls** заменяет их вопросительными знаками в этой локали. Кроме того, попытка отправить почту с такими символами из Mutt или Pine приводит к тому что, отправляемые сообщения не соответствуют RFC (кодировка в исходящей почте указана как `unknown 8-bit`). Рекомендуется использовать локаль `C`, если вы уверены, что вам никогда не понадобятся 8-битные символы.

9.8. Создание файла /etc/inputrc

Файл `inputrc` это конфигурационный файл библиотеки readline, который предоставляет возможности редактирования, когда пользователь вводит строку с терминала. Он работает путем преобразования ввода с клавиатуры в определенные действия. Readline используется bash и большинством других оболочек, а также многими другими приложениями.

Большинство людей не нуждаются в специальных настройках, поэтому приведенная ниже команда создает глобальный `/etc/inputrc`, используемый всеми, кто входит в систему. Если позже вы решите, что вам нужно переопределить значения по умолчанию для одного из пользователей, вы можете создать файл `.inputrc` в домашнем каталоге пользователя и указать в нём измененные настройки.

Дополнительные сведения о редактировании файла `inputrc` см. в разделе **info bash** в секции *Readline Init File*. Также хорошим источником информации является **info readline**.

Ниже приведен общий глобальный `inputrc` с комментариями, объясняющими, что делают различные параметры. Обратите внимание, что комментарии не могут находиться в той же строке, что и команды. Создайте файл с помощью следующей команды:

```
cat > /etc/inputrc << "EOF"
# Begin /etc/inputrc
# Modified by Chris Lynn <roryo@roryo.dynup.net>

# Разрешить перенос приглашения оболочки на следующую строку
set horizontal-scroll-mode Off

# Разрешить 8-битный ввод
set meta-flag On
set input-meta On

# Отключить обрезание восьмого бита в вводимых символах
set convert-meta Off

# Выводить на экран все восемь бит
set output-meta On

# Звуковой сигнал - none, visible или audible, соответственно никакого, видимый или слышимый
set bell-style none

# Нижеследующие команды привязывают escape-последовательности (первый аргумент)
# к специфичным для Readline функциям (второй аргумент)
"\eOd": backward-word
"\eOc": forward-word

# для консоли linux
"\e[1~": beginning-of-line
"\e[4~": end-of-line
"\e[5~": beginning-of-history
"\e[6~": end-of-history
"\e[3~": delete-char
"\e[2~": quoted-insert

# для xterm
"\eOH": beginning-of-line
"\eOF": end-of-line

# для Konsole
"\e[H": beginning-of-line
"\e[F": end-of-line

# End /etc/inputrc
EOF
```

9.9. Создание файла /etc/shells

Файл `shells` содержит список оболочек входа в систему. Приложения используют этот файл для определения корректности оболочки. Для каждой оболочки должна присутствовать одна строка, состоящая из пути к файлу оболочки относительно корня структуры каталогов (/).

Например, **chsh** обращается к этому файлу, чтобы определить, может ли непривилегированный пользователь изменить оболочку входа для своей учетной записи. Если имя команды не указано в списке, пользователю будет отказано в возможности изменять оболочки.

Это обязательное условие для таких приложений, как GDM, которые не заполняют список пользователей, если ему не удаётся найти `/etc/shells` , или демонов FTP, которые традиционно запрещают доступ пользователям с оболочками, не включенными в этот файл.

```
cat > /etc/shells << "EOF"
# Begin /etc/shells

/bin/sh
/bin/bash

# End /etc/shells
EOF
```

Глава 10. Делаем систему LFS загрузочной

10.1. Введение

Пришло время сделать систему LFS загрузочной. В этой главе обсуждается создание файла `/etc/fstab`, сборка ядра для новой системы и установка загрузчика GRUB, чтобы система LFS могла быть выбрана для загрузки при запуске.

10.2. Создание файла /etc/fstab

Файл `/etc/fstab` используется некоторыми программами для определения того, какие файловые системы должны монтироваться по умолчанию, в каком порядке и какие из них должны быть проверены (на наличие ошибок целостности) перед монтированием. Создайте новую таблицу файловых систем следующим образом:

```
cat > /etc/fstab << "EOF"
# Begin /etc/fstab

# file system  mount-point  type     options             dump  fsck
#                                                              order

/dev/<xxx>     /            <fff>    defaults            1     1
/dev/<yyy>     swap         swap     pri=1               0     0
proc           /proc        proc     nosuid,noexec,nodev 0     0
sysfs          /sys         sysfs    nosuid,noexec,nodev 0     0
devpts         /dev/pts     devpts   gid=5,mode=620      0     0
tmpfs          /run         tmpfs    defaults            0     0
devtmpfs       /dev         devtmpfs mode=0755,nosuid    0     0
tmpfs          /dev/shm     tmpfs    nosuid,nodev        0     0
cgroup2        /sys/fs/cgroup cgroup2  nosuid,noexec,nodev 0     0

# End /etc/fstab
EOF
```

Замените `<xxx>`, `<yyy>`, и `<fff>` подходящими для системы значениями, например, `sda2`, `sda5`, и `ext4`. Для получения подробной информации о параметрах в этом файле, смотрите *fstab(5)*.

Файловым системам операционных систем MS DOS и Windows (таким как vfat, ntfs, smbfs, cifs, iso9660, udf) требуется специальная опция utf8, чтобы не-ASCII символы в именах файлов интерпретировались правильно. Для локалей, отличных от UTF-8, значение `iocharset` должно быть таким же, как набор символов локали и настроено так, чтобы ядро понимало его. Это будет работать, если соответствующее определение набора символов (находится в разделе File systems -> Native Language Support при настройке ядра) было скомпилировано в ядро или собрано как модуль. Однако, если набор символов локали — UTF-8, параметр `iocharset=utf8` сделает файловую систему чувствительной к регистру. Чтобы исправить это, используйте специальную опцию `utf8` вместо `iocharset=utf8` для локалей UTF-8. Параметр «codepage» также необходим для файловых систем vfat и smbfs. Он должен быть установлен на номер кодовой страницы, используемый в MS-DOS в вашей стране. Например, для монтирования флешек пользователь локали ru_RU.KOI8-R должен установить следующие значения в группе параметров строки монтирования в `/etc/fstab`:

```
noauto,user,quiet,showexec,codepage=866,iocharset=koi8r
```

Соответствующий фрагмент параметров для пользователей ru_RU.UTF-8 выглядит следующим образом:

```
noauto,user,quiet,showexec,codepage=866,utf8
```

Обратите внимание, что `iocharset` используется по умолчанию для `iso8859-1` (которая сохраняет файловую систему нечувствительной к регистру), а параметр `utf8` указывает ядру, что нужно преобразовать имена файлов с использованием UTF-8, чтобы их можно было интерпретировать в локали UTF-8.

Также возможно указать значения кодовой страницы по умолчанию и iocharset для некоторых файловых систем во время настройки ядра. Соответствующие параметры называются «Default NLS Option» (`CONFIG_NLS_DEFAULT`), «Default Remote NLS Option» (`CONFIG_SMB_NLS_DEFAULT`), «Default codepage for FAT» (`CONFIG_FAT_DEFAULT_CODEPAGE`) и «Default iocharset for FAT» (`CONFIG_FAT_DEFAULT_IOCHARSET`). Нет возможности указать эти параметры для файловой системы ntfs во время компиляции ядра.

Для некоторых типов жестких дисков можно сделать файловую систему ext3 более устойчивой к сбоям питания. Чтобы сделать это, добавьте параметр `barrier=1` к соответствующей записи в `/etc/fstab`. Чтобы проверить, поддерживает ли диск эту опцию, запустите *hdparm* на соответствуюшем разделе. Например, если:

```
hdparm -I /dev/sda | grep NCQ
```

возвращает непустой вывод, опция поддерживается.

Примечание: разделы на основе управления логическими томами (LVM) не могут использовать параметр `barrier`.

10.3. Linux-6.10.5

Этот пакет содержит ядро Linux.

Приблизительное время сборки: 0.4 - 32 SBU (обычно около 2.5 SBU)

Требуемое дисковое пространство: 1.7 - 14 GB (обычно около 2.3 GB)

10.3.1. Установка ядра

Сборка ядра состоит из нескольких этапов — настройка, компиляция и установка. Ознакомьтесь с файлом `README` в дереве исходных текстов, чтобы узнать об альтернативных способах настройки ядра.

Важно

Сборка ядра Linux в первый раз — одна из самых сложных задач в LFS. Правильный выбор параметров зависит от конкретного оборудования для целевой системы и ваших потребностей. Для ядра доступно почти 12 000 элементов конфигурации, хотя для большинства компьютеров требуется только около трети из них. Редакторы LFS рекомендуют пользователям, не знакомым с этим процессом, внимательно следовать описанным ниже процедурам. Главная цель сейчас состоит в том, чтобы довести первоначальную систему до состояния, когда вы сможете войти в систему из командной строки при последующей перезагрузке в Раздел 11.3, «Перезагрузка системы». Вопросы оптимизации и кастомизации второстепенны.

Для получения общей информации о конфигурации ядра смотрите *https://mirror.linuxfromscratch.ru/hints/downloads/files/kernel-configuration.txt*. Дополнительную информацию о настройке и сборке ядра можно найти по адресу *https://anduin.linuxfromscratch.org/LFS/kernel-nutshell/*. Эти ссылки немного устарели, но все же дают разумное представление о процессе.

Если ничего не помогает, вы можете обратиться за помощью в список рассылки *lfs-support*. Обратите внимание, что подписка необходима для того, чтобы рассылка не содержала спама.

Подготовьте пакет к компиляции, выполнив следующую команду:

```
make mrproper
```

Выполнение этой команды гарантирует, что дерево исходников будет абсолютно чистым. Разработчики ядра рекомендуют запускать эту команду перед каждой компиляцией. Не следует полагаться на то, что дерево исходных текстов ядра будет чистым после распаковки.

Существует несколько способов настройки параметров ядра. Обычно это делается с помощью псевдографического интерфейса, например так:

```
make menuconfig
```

Значения необязательных переменных окружения make:

`LANG=<переменная_LANG_хоста> LC_ALL=`

Устанавливает значение локали на то, которое используется на хосте. Это может понадобиться для правильного отображения интерфейса menuconfig с помощью ncurses в текстовой консоли Linux с UTF-8.

Если это необходимо, обязательно замените значение *`<переменной_LANG>`* на значение переменной `$LANG` вашего хоста. В качестве альтернативы вы можете использовать значения переменных `$LC_ALL` или `$LC_CTYPE` .

make menuconfig

Эта команда запускает интерфейс на основе ncurses. Для использования других (графических) интерфейсов, выполните **make help**.

Примечание

Хорошей отправной точкой для настройки ядра, может стать запуск команды **make defconfig**. В результате её выполнения будет создана базовая конфигурация с учётом архитектуры системы.

Обязательно включите/отключите/настройте следующие параметры, иначе система может работать некорректно или вообще не загружаться:

```
General setup --->
  [ ] Compile the kernel with warnings as errors                        [WERROR]
  CPU/Task time and stats accounting --->
    [*] Pressure stall information tracking                               [PSI]
    [ ]   Require boot parameter to enable pressure stall information tracking
                                                          ...  [PSI_DEFAULT_DISABLED]
  < > Enable kernel headers through /sys/kernel/kheaders.tar.xz      [IKHEADERS]
  [*] Control Group support --->                                       [CGROUPS]
    [*] Memory controller                                                [MEMCG]
  [ ] Configure standard kernel features (expert users) --->           [EXPERT]

Processor type and features --->
  [*] Build a relocatable kernel                                   [RELOCATABLE]
  [*]   Randomize the address of the kernel image (KASLR)       [RANDOMIZE_BASE]

General architecture-dependent options --->
  [*] Stack Protector buffer overflow detection                 [STACKPROTECTOR]
  [*]   Strong Stack Protector                               [STACKPROTECTOR_STRONG]

Device Drivers --->
  Generic Driver Options --->
    [ ] Support for uevent helper                                 [UEVENT_HELPER]
    [*] Maintain a devtmpfs filesystem to mount at /dev              [DEVTMPFS]
    [*]   Automount devtmpfs at /dev, after the kernel mounted the rootfs
                                                             ...  [DEVTMPFS_MOUNT]
  Graphics support --->
    < /*/M> Direct Rendering Manager (XFree86 4.1.0 and higher DRI support) --->
                                                                    ...  [DRM]
      # If [DRM] is selected as * or M, this must be selected:
      [ /*] Enable legacy fbdev support for your modesetting driver
                                                          ...  [DRM_FBDEV_EMULATION]
    Console display driver support --->
      # If [DRM] is selected as * or M, this must be selected:
      [ /*] Framebuffer Console support                    [FRAMEBUFFER_CONSOLE]
```

Включите некоторые дополнительные функции, если вы собираете 64-битную систему. Если вы используете menuconfig, включите их в следующем порядке: сначала *CONFIG_PCI_MSI* , затем *CONFIG_IRQ_REMAP* , и, наконец, *CONFIG_X86_X2APIC* , потому что параметр отображается только после выбора его зависимости.

```
Processor type and features --->
  [*] Support x2apic                                               [X86_X2APIC]

Device Drivers --->
  [*] PCI support --->                                                    [PCI]
    [*] Message Signaled Interrupts (MSI and MSI-X)                    [PCI_MSI]
  [*] IOMMU Hardware Support --->                                  [IOMMU_SUPPORT]
    [*] Support for Interrupt Remapping                               [IRQ_REMAP]
```

Если вы создаете 32-разрядную систему, работающую на оборудовании с объемом оперативной памяти более 4 ГБ, измените конфигурацию таким образом, чтобы ядро могло использовать до 64 ГБ оперативной памяти:

```
Processor type and features --->
  High Memory Support --->
    (X) 64GB                                                [HIGHMEM64G]
```

Если раздел для системы LFS находится на NVME SSD (то есть узлом устройства для раздела является `/dev/nvme*` , а не `/dev/sd*`), включите параметр NVME support, иначе система LFS не будет загружаться:

```
Device Drivers --->
  NVME Support --->
    <*> NVM Express block device                          [BLK_DEV_NVME]
```

Есть несколько других параметров, которые могут понадобиться в зависимости от особенностей системы. Для получения списка необходимых опций для пакетов BLFS смотрите *Список опций ядра BLFS*.

Примечание

Если ваша хост поддерживает UEFI и вы хотите загрузить LFS с помощью него, вам необходимо настроить некоторые параметры ядра, следуя инструкции *на странице BLFS*, **даже если вы будете использовать загрузчик UEFI из основного дистрибутива**.

Примечание

Ядро в multilib-системе должно иметь возможность идентифицировать и запускать двоичные файлы, скомпилированные для архитектур, отличных от архитектуры, используемой по умолчанию.

Если была добавлена поддержка любого 32-битного ABI, убедитесь, что выбран параметр "IA32 Emulation". Параметр 'IA32 a.out support' не является обязательным и включен как модуль.

```
Binary Emulations  --->
    [*] IA32 Emulation [CONFIG_IA32_EMULATION]
    <M>   IA32 a.out support [CONFIG_IA32_AOUT]
```

Пояснения для выбранных выше параметров ядра:

`Randomize the address of the kernel image (KASLR)`
Включите ASLR для образа ядра, чтобы уменьшить вероятность некоторых атак, основанных на фиксированных адресах конфиденциальных данных или кода в ядре.

`Compile the kernel with warnings as errors`
Включение этого параметра может привести к сбою сборки, если компилятор и/или конфигурация отличается от конфигурации ядра разработчиков.

`Enable kernel headers through /sys/kernel/kheaders.tar.xz`
Для сборки ядра с этим параметром необходим пакет **cpio**. **cpio** не устанавливается в LFS.

`Configure standard kernel features (expert users)`
Эта опция приведет к отображению некоторых параметров в интерфейсе конфигурации, но изменение этих параметров может быть опасным. Не используйте её, если вы не знаете, что делаете.

`Strong Stack Protector`
Включите SSP для ядра. Мы включили его для всего пользовательского пространства с помощью `--enable-default-ssp` , настроив GCC, но ядро не использует настройки GCC по умолчанию для SSP. Мы включаем это явно здесь.

Support for uevent helper
: Включение этого параметра может вызвать сбои при управление устройствами через Udev.

Maintain a devtmpfs
: С помощью этого параметра узлы устройств создаются автоматически и заполняются самим ядром, даже без запуска Udev. Udev будет работать поверх, управляя разрешениями и добавляя необходимые символические ссылки. Этот элемент конфигурации необходим всем пользователям Udev.

Automount devtmpfs at /dev
: Этот параметр позволит смонтировать представление ядра устройств в /dev при переключении на корневую файловую систему непосредственно перед запуском init.

Enable legacy fbdev support for your modesetting driver и *Framebuffer Console support*
: Они необходимы для отображения консоли Linux на графическом процессоре, управляемом драйвером DRI (Direct Rendering Infrastructure - инфраструктура прямого рендеринга). Если `CONFIG_DRM` (Direct Rendering Manager - диспетчер прямого рендеринга) включен, вам также следует включить эти две опции, иначе вы увидите пустой экран после загрузки драйвера DRI.

Support x2apic
: Поддержка запуска 64-разрядного контроллера прерываний для x86 процессоров в режиме x2APIC. x2APIC может быть включен в BIOS на системах x86 и у ядра собранного без этой опции будет kernel panic при загрузке. Эта опция не окажет никакого эффекта, но и не причиняет вреда, если x2APIC отключен в BIOS.

В качестве альтернативы, в некоторых ситуациях может быть уместно использование команды **make oldconfig**. Смотрите файл `README` для получения дополнительной информации.

По желанию, вы можете пропустить настройку ядра, скопировав конфигурационный файл ядра `.config`, из хост системы(если он доступен) в каталог куда было распаковано ядро `linux-6.10.5`. Однако, мы не рекомендуем этот вариант. Намного лучше изучить все параметры меню и создать конфигурацию ядра с нуля.

Скомпилируйте образ ядра и модули:

```
make
```

При использовании модулей, могут потребоваться файлы конфигурации, которые расположены в каталоге `/etc/modprobe.d`. Информация о модулях и конфигурации ядра находится в Раздел 9.3, «Взаимодействие с устройствами и модулями» и в документации к ядру `linux-6.10.5/Documentation`. Кроме этого, стоит ознакомиться с руководством *modprobe.d(5)*.

Если поддержка модулей не была отключена в параметрах ядра, установите модули с помощью:

```
make modules_install
```

После окончания компиляции, необходимо выполнить еще несколько шагов для завершения установки ядра. Некоторые файлы должны быть скопированы в каталог `/boot`.

Внимание

Если вы решили использовать отдельный `/boot` раздел для системы LFS (возможно, общий раздел `/boot` с хост-дистрибутивом), скопированные ниже файлы должны быть помещены туда. Самый простой способ сделать это — сначала создать запись для `/boot` в `/etc/fstab` (подробности читайте в предыдущем разделе), затем выполните следующую команду от имени пользователя `root` в среде *chroot*:

```
mount /boot
```

Путь к узлу устройства в команде опущен, поскольку **mount** может прочитать его из `/etc/fstab`.

Путь к образу ядра может различаться в зависимости от используемой платформы. Имя файла, может быть произвольным, но начинаться должно с *vmlinuz* для обеспечения совместимости с автоматической настройкой процесса загрузки, описанного в следующем разделе. Следующая команда предполагает архитектуру x86:

```
cp -iv arch/x86/boot/bzImage /boot/vmlinuz-6.10.5-lfs-12.2
```

`System.map` - это символьный файл для ядра. Он содержит точки входа каждой функции в API ядра, а также адреса структур данных для запущенного ядра. Он используется в качестве ресурса при исследовании проблем с ядром. Выполните следующую команду для установки файла:

```
cp -iv System.map /boot/System.map-6.10.5
```

Файл конфигурации ядра `.config` создается на шаге **make menuconfig** и содержит все параметры ядра, которое было скомпилировано только что. Рекомендуется сохранить этот файл на будущее:

```
cp -iv .config /boot/config-6.10.5
```

Установите документацию ядра:

```
cp -r Documentation -T /usr/share/doc/linux-6.10.5
```

Важно отметить, что файлы в каталоге с исходным кодом ядра не принадлежат пользователю *root*. Всякий раз, когда пакет распаковывается от имени пользователя *root* (как это и выполнялось внутри среды chroot), файлы имеют те идентификаторы пользователя и группы, которые были присвоены при распаковке. Обычно это не вызывает проблем для других устанавливаемых пакетов, так как каталог с исходниками удаляется после установки пакета. Однако исходный код ядра Linux часто сохраняется в течение длительного времени. Из-за этого существует вероятность того, что идентификатор пользователя, используемый при распаковке, будет назначен другому пользователю. В таком случае, этот пользователь будет иметь доступ на запись в этот каталог.

Примечание

В ряде случаев требуется обновить конфигурацию ядра для пакетов, которые будут установлены позже в BLFS. В отличии от других пакетов, нет необходимости удалять дерево исходного кода ядра после установки только что собранного ядра.

Если вы планируете оставить каталог с исходным кодом ядра, выполните команду **chown -R 0:0 *** находясь в каталоге `linux-6.10.5`, чтобы все файлы принадлежали пользователю *root*.

Предупреждение

Иногда, в документации к ядру, рекомендуют создать символическую ссылку `/usr/src/linux` указывающую на каталог с исходниками ядра. Эта рекомендация относится к ядрам до версии 2.6 и *не должна* выполняться в системе LFS, так как это может вызвать проблемы с пакетами, которые вы, возможно, захотите собрать, когда ваша базовая система LFS будет готова.

Предупреждение

Заголовочные файлы в системном каталоге `include` (`/usr/include`) *всегда* используются те, которые применялись при компиляции Glibc, то есть подготовленные заголовочные файлы, установленные в Раздел 5.4, «Заголовочные файлы Linux-6.10.5 API». Поэтому их *никогда* не следует заменять на чистые заголовочные файлы ядра или любые другие подготовленные заголовочные файлы.

10.3.2. Настройка порядка загрузки модулей Linux

В большинстве случаев модули Linux загружаются автоматически, но иногда требуется определенный порядок. Программа, которая загружает модули, **modprobe** или **insmod**, использует файл `/etc/modprobe.d/usb.conf` как раз для этой цели. Этот файл должен быть заполнен таким образом, что если USB-драйверы (ehci_hcd, ohci_hcd и uhci_hcd) были собраны в виде модулей, то они будут загружены в правильном порядке; ehci_hcd должен быть загружен до ohci_hcd и uhci_hcd для того, чтобы избежать предупреждений во время загрузки.

Создайте новый файл `/etc/modprobe.d/usb.conf`, выполнив следующую команду:

```
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
```

10.3.3. Содержимое пакета Linux

Установленные файлы: config-6.10.5, vmlinuz-6.10.5-lfs-12.2, и System.map-6.10.5
Созданные каталоги: /lib/modules, /usr/share/doc/linux-6.10.5

Краткое описание

`config-6.10.5` — Содержит в себе все параметры конфигурации ядра

`vmlinuz-6.10.5-lfs-12.2` — Ядро системы Linux. При включении компьютера ядро — это первая загружаемая часть операционной системы. Оно обнаруживает и инициализирует все компоненты аппаратного обеспечения компьютера, делает их доступными в виде дерева каталогов с файлами для доступа к ним программ и превращает один процессор в мультизадачную машину, способную выполнять множество программ как будто одновременно.

`System.map-6.10.5` — Список адресов и символов; файл содержит точки входа и адреса всех функций и структур данных в ядре

10.4. Использование GRUB для настройки процесса загрузки

Примечание

Если ваша система поддерживает UEFI и вы хотите загрузить LFS с помощью UEFI, вам следует пропустить инструкции на этой странице, но все равно изучить синтаксис `grub.cfg` и способ указания раздела в файле с этой страницы, а также настроить GRUB с поддержкой UEFI используя инструкции, приведенные на *странице BLFS*.

10.4.1. Введение

Предупреждение

При неправильной настройке GRUB ваша система перестанет загружаться без вспомогательно загрузочного CD-ROM или USB-накопителя. Для загрузки системы LFS этот раздел необязателен. Вы можете просто использовать существующий загрузчик, например Grub-Legacy, GRUB2 или LILO.

Убедитесь, что аварийный загрузочный диск готов к «спасению» компьютера, если он перестанет загружаться. Если у вас еще нет загрузочного диска, вы можете создать его. Для этого необходимо перейти в раздел BLFS и установить программу **`xorriso`** из пакета *libisoburn*.

```
cd /tmp
grub-mkrescue --output=grub-img.iso
xorriso -as cdrecord -v dev=/dev/cdrw blank=as_needed grub-img.iso
```

10.4.2. Соглашения об именовании GRUB

GRUB использует собственную структуру именования дисков и разделов в виде *(hdn,m)*, где *n* — номер жесткого диска, а *m* — номер раздела. Номера жестких дисков начинаются с нуля, а номера разделов начинаются с единицы для обычных разделов (с пяти для расширенных разделов). Обратите внимание, что это отличается от более ранних версий, где оба номера начинались с нуля. Например, раздел `sda1` это *(hd0,1)* в GRUB, а `sdb3` *(hd1,3)*. В отличие от Linux, GRUB не считает приводы CD-ROM жесткими дисками. Например, если используемый CD-привода определяется как `hdb`, а второй жесткий диск как `hdc`, этот второй жесткий диск все равно будет *(hd1)*.

10.4.3. Настройка

GRUB записывает данные на первый физический сектор жесткого диска. Эта область не является частью какой-либо файловой системы. Программа в загрузочном разделе имеет доступ к модулям GRUB расположенным по умолчанию в /boot/grub/.

Расположение загрузочного раздела - это выбор пользователя, который влияет на конфигурацию. Одна из рекомендаций заключается в том, чтобы иметь отдельный небольшой раздел (примерно 200 МБ) исключительно для загрузочной информации. В этом случае каждая сборка, будь то LFS или другой дистрибутив, может обращаться к тем же загрузочным файлам, а доступ может быть получен из любой загруженной системы. Если вы решите так сделать, вам необходимо примонтировать отдельный раздел, переместить все файлы из текущего каталога `/boot` (например, ядро Linux, которое вы создали на предыдущем этапе) в новый раздел. Затем нужно отмонтировать раздел и примонтировать его заново в каталог `/boot` . Когда вы это сделаете, обязательно обновите данные в файле `/etc/fstab` .

Оставить `/boot` на текущем разделе LFS это тоже рабочее решение, но его настройка для загрузки нескольких систем сложнее.

Используя информацию выше, определите соответствующие точки монтирования для корневого раздела (или загрузочного раздела, если используется отдельный). В следующем примере предполагается, что корневым (или отдельным загрузочным) разделом является `sda2`.

Установите файлы GRUB в каталог `/boot/grub` и настройте загрузочный сектор:

Предупреждение

Следующая команда перезапишет текущий загрузчик. Не выполняйте эту команду, если это не нужно, например, если вы используете сторонний менеджер загрузки для управления главной загрузочной записью (MBR).

```
grub-install /dev/sda
```

Примечание

Если система была загружена с использованием UEFI, **grub-install** попытается установить файлы для *x86_64-efi*, но эти файлы не были установлены в Глава 8. Если это так, добавьте `--target i386-pc` к приведенной выше команде.

10.4.4. Создание файла конфигурации GRUB

Создайте файл `/boot/grub/grub.cfg` :

```
cat > /boot/grub/grub.cfg << "EOF"
# Begin /boot/grub/grub.cfg
set default=0
set timeout=5

insmod part_gpt
insmod ext2
set root=(hd0,2)

menuentry "GNU/Linux, Linux 6.10.5-lfs-12.2" {
        linux   /boot/vmlinuz-6.10.5-lfs-12.2 root=/dev/sda2 ro
}
EOF
```

Команды **insmod** загружают модули GRUB с именами `part_gpt` и `ext2`. Несмотря на название, `ext2` фактически поддерживаются файловые системы `ext2`, `ext3` и `ext4`. Команда **grub-install** встроила некоторые модули в основной образ GRUB (установленный в MBR или раздел GRUB BIOS) для доступа к другим модулям (в `/boot/grub/i386-pc`), поэтому в стандартной конфигурации эти два модуля уже встроены и эти две команды **insmod** ничего не будут делать. В любом случае, от них нет никакого вреда, но они могут понадобиться в некоторых редких конфигурациях.

Примечание

С точки зрения GRUB, файлы ядра относятся к используемому разделу. Если вы используется отдельный раздел /boot, удалите /boot из приведенной выше строки *linux*. Вам также потребуется изменить строку *set root* так, чтобы она указывала на загрузочный раздел.

Примечание

Наименование раздела для GRUB может измениться, если вы добавили или удалили некоторые диски (это могут быть как съемные диски, так и USB-устройства). Изменение может привести к сбою загрузки, потому что `grub.cfg` ссылается на «старые» указатели. Чтобы не столкнуться с этой проблемой, необходимо использовать UUID раздела и файловой системы вместо указателя GRUB для указания устройства. Запустите команду **lsblk -o UUID,PARTUUID,PATH,MOUNTPOINT**, чтобы посмотреть UUID ваших файловых систем (в столбце `UUID`) и разделов (в столбце `PARTUUID`). Затем замените `set root=(hdx,y)` на `search --set=root --fs-uuid <UUID файловой системы, в которой установлено ядро>`, и замените `root=/dev/sda2` на `root=PARTUUID=<UUID раздела, в котором собрана LFS>`.

Обратите внимание, что UUID раздела и UUID файловой системы на этом разделе это совершенно разные вещи. Некоторые онлайн-ресурсы могут предлагать вам использовать `root=UUID=<UUID файловой системы>` вместо `root=PARTUUID=<UUID раздела>`, но для этого требуется initramfs, которая не рассматривается в LFS.

Имя узла устройства для раздела в `/dev` также может измениться (хотя это менее вероятно, чем изменение указателя GRUB). Вы можете заменить пути к узлам устройств, таким как `/dev/sda1` на `PARTUUID=<UUID раздела>`, в `/etc/fstab` , чтобы избежать потенциального сбоя загрузки в случае, если имя узла устройства изменилось.

GRUB - чрезвычайно мощная программа, предоставляющая огромное количество вариантов загрузки с самых разных устройств, работающих систем и типов разделов. Существует также множество опций настройки, таких как графические экраны-заставки, воспроизведение звука, ввод с помощью мыши и т. д., детали этих опций выходят за рамки этой инструкции.

Внимание

Существует команда grub-mkconfig, которая может автоматически записывать файл конфигурации. Она использует набор скриптов из каталога /etc/grub.d/ и уничтожит любые сделанные вами настройки. Эти скрипты предназначены в первую очередь для обычных дистрибутивов и не рекомендуются для LFS. Если вы устанавливаете коммерческий дистрибутив Linux, есть вероятность, что эта программа будет запущена. Обязательно создайте резервную копию файла grub.cfg.

Глава 11. Заключение

11.1. Заключение

Отлично! Новая система LFS установлена! Желаем успехов в работе с вашей новой, блестящей, самостоятельно собранной Linux системой.

Может быть хорошей идеей создать файл `/etc/lfs-release` . Имея этот файл, вам (и нам, если вам в какой-то момент понадобится обратиться за помощью) будет проще узнать, какая версия LFS установлена в системе. Создайте этот файл, выполнив следующую команду:

```
echo 12.2 > /etc/lfs-release
```

Следующие два файла, содержащие описание установленной системы, могут использоваться пакетами, устанавливаемыми позже, либо в бинарном виде, либо путем их сборки.

Первый показывает статус вашей новой системы по отношению к стандарту LSB. Чтобы создать этот файл, выполните:

```
cat > /etc/lsb-release << "EOF"
DISTRIB_ID="Linux From Scratch"
DISTRIB_RELEASE="12.2"
DISTRIB_CODENAME="<your name here>"
DISTRIB_DESCRIPTION="Linux From Scratch"
EOF
```

Второй содержит примерно ту же информацию и используется systemd и некоторыми графическими средами рабочего стола. Чтобы создать этот файл, выполните:

```
cat > /etc/os-release << "EOF"
NAME="Linux From Scratch"
VERSION="12.2"
ID=lfs
PRETTY_NAME="Linux From Scratch 12.2"
VERSION_CODENAME="<your name here>"
HOME_URL="https://linuxfromscratch.ru/"
EOF
```

Обязательно настройте значения 'DISTRIB_CODENAME' и 'VERSION_CODENAME', чтобы сделать название вашей новой системы уникальным.

11.2. Вступите в ряды пользователей LFS

Теперь, когда вы закончили изучение книги LFS, хотите добавить себя в список пользователей LFS? Перейдите по ссылке *https://www.linuxfromscratch.org/cgi-bin/lfscounter.php* и зарегистрируйтесь. Введите ваше имя и версию LFS, которую вы использовали.

Давайте выполним перезагрузку в систему LFS.

11.3. Перезагрузка системы

Теперь, когда все программное обеспечение установлено, пришло время перезагрузить ваш компьютер. Однако есть несколько вещей, которые нужно проверить. Вот некоторые предложения:

- Установить *прошивки*, если они необходимы для правильной работы вашего оборудования.
- Убедитесь, что установлен пароль для пользователя `root`.

- На данном этапе также уместно ознакомиться со следующими конфигурационными файлами.
  - /etc/bashrc
  - /etc/dircolors
  - /etc/fstab
  - /etc/hosts
  - /etc/inputrc
  - /etc/profile
  - /etc/resolv.conf
  - /etc/vimrc
  - /root/.bash_profile
  - /root/.bashrc
  - /etc/sysconfig/ifconfig.eth0

Теперь, после всего, давайте перейдём к первой загрузке нашей новой системы LFS. Для начала, выйдем из chroot-окружения:

```
logout
```

Затем размонтируйте виртуальные файловые системы:

```
umount -v $LFS/dev/pts
mountpoint -q $LFS/dev/shm && umount -v $LFS/dev/shm
umount -v $LFS/dev
umount -v $LFS/run
umount -v $LFS/proc
umount -v $LFS/sys
```

Если было создано несколько разделов, размонтируйте их перед размонтированием основного, вот так:

```
umount -v $LFS/home
umount -v $LFS/usr
```

Размонтируйте саму файловую систему LFS:

```
umount -v $LFS
```

Теперь, выполните перезагрузку системы.

Предполагается, что загрузчик GRUB был настроен ранее, поэтому пункт меню *LFS 12.2* будет загружен автоматически.

После завершения перезагрузки, система LFS будет готова к использованию. Вы увидите простую подсказку «login: ». На этом этапе вы можете перейти к *книге BLFS*, где вы установите дополнительное программное обеспечение в соответствии с вашими потребностями.

Если перезагрузка завершилась **неудачей**, самое время устранить эти неполадки. Советы по решению проблем с начальной загрузкой, смотрите на странице *https://mirror.linuxfromscratch.ru/lfs/troubleshooting.html*.

11.4. Дополнительные ресурсы

Благодарим за прочтение книги LFS. Мы надеемся, что эта книга была полезна и вы узнали больше о процессе создания системы с нуля.

Теперь, когда система LFS установлена, вы можете задаться вопросом «Что дальше?» Чтобы ответить на этот вопрос, мы составили для вас список ресурсов.

- Обслуживание

  Для всего программного обеспечения регулярно появляются сообщения об ошибках и уведомления безопасности. Поскольку система LFS компилируется из исходного кода, вы должны быть в курсе таких отчетов. Существует несколько онлайн-ресурсов, которые отслеживают такие отчеты, некоторые из них приведены ниже:

  - *Рекомендации по безопасности LFS*

    Это список уязвимостей системы безопасности, обнаруженных в книге LFS после ее публикации.

  - *Список рассылки по безопасности ПО с открытым исходным кодом*

    Это список рассылки для обсуждения недостатков безопасности, концепций и практик в сообществе Open Source.

- Советы LFS

  Советы LFS представляют собой коллекцию обучающих материалов, собранную добровольцами сообщества LFS. Советы доступны по адресу *https://mirror.linuxfromscratch.ru/hints/downloads/files/*.

- Списки рассылки

  Существует несколько списков рассылки LFS, на которые вы можете подписаться, если нуждаетесь в помощи, хотите быть в курсе последних событий, хотите внести свой вклад в проект и многое другое. Посетите Глава 1 - Списки рассылки для получения дополнительной информации.

- Проект документации по Linux (TLDP)

  Целью проекта TLDP является сотрудничество по всем вопросам связанным с документацией по Linux. TLDP содержит большую коллекцию инструкций, руководств и справочных страниц. Она расположена по адресу *https://tldp.org/*.

11.5. Начало работы после сборки LFS

11.5.1. Что делать дальше?

Теперь, когда LFS собрана и у вас есть загружаемая система, необходимо решить, что же делать дальше? Следующий шаг - определиться, как использовать систему. Как правило, следует учитывать две широкие категории: рабочая станция или сервер. Действительно, эти категории не являются взаимоисключающими. Приложения, необходимые для каждой категории, можно объединить в одну систему, но пока давайте рассмотрим их по отдельности.

Сервер — более простая категория. Как правило, это веб-сервер, такой как *Apache*, и сервер баз данных, например, *MariaDB*. Однако возможны и другие варианты. К этой же категории относятся операционные системы для встраиваемых устройств.

Рабочая станция же, гораздо сложнее. Обычно для нее требуется среда рабочего стола, например, *LXDE*, *XFCE*, *KDE*, или *Gnome* основанные на базовом *графическом окружении* и наборе графических приложений, таких как *веб-браузер Firefox*, *почтовый клиент Thunderbird*, или *пакет офисных приложений LibreOffice*. Для этих приложений требуется множество (может быть несколько сотен, в зависимости от ваших потребностей) пакетов вспомогательных приложений и библиотек.

В дополнение к вышесказанному, существует набор приложений для управления системой. Все эти приложения есть в справочнике BLFS, но не все пакеты необходимы в каждом конкретном окружении. Например *клиент dhcpcd*, обычно не требуется на серверах или *управление параметрами беспроводных сетей* - полезно только для ноутбуков и других портативных систем.

11.5.2. Работа в базовой среде LFS

Когда вы впервые загружаетесь в LFS, у вас есть все необходимые инструменты для сборки дополнительных пакетов. К сожалению, набор программ пользовательского окружения довольно скудный. Есть несколько способов исправить это:

11.5.2.1. Работа с хоста LFS в chroot

Этот метод обеспечивает полноценную графическую среду, в которой доступен полнофункциональный браузер и возможности копирования/вставки. Также он позволяет использовать приложения хоста, такие как wget, для загрузки исходных текстов пакетов в каталог, доступный при работе в среде chroot.

Чтобы правильно собрать пакеты в chroot, вам необходимо не забыть смонтировать виртуальные файловые системы, если они еще не смонтированы. Один из способов сделать это — создать скрипт в **ХОСТОВОЙ** системе:

```
cat > ~/mount-virt.sh << "EOF"
#!/bin/bash

function mountbind
{
   if ! mountpoint $LFS/$1 >/dev/null; then
     $SUDO mount --bind /$1 $LFS/$1
     echo $LFS/$1 mounted
   else
     echo $LFS/$1 already mounted
   fi
}

function mounttype
{
   if ! mountpoint $LFS/$1 >/dev/null; then
     $SUDO mount -t $2 $3 $4 $5 $LFS/$1
     echo $LFS/$1 mounted
   else
     echo $LFS/$1 already mounted
   fi
}

if [ $EUID -ne 0 ]; then
  SUDO=sudo
else
  SUDO=""
fi

if [ x$LFS == x ]; then
  echo "LFS not set"
  exit 1
fi

mountbind dev
mounttype dev/pts devpts devpts -o gid=5,mode=620
mounttype proc    proc   proc
mounttype sys     sysfs  sysfs
mounttype run     tmpfs  run
if [ -h $LFS/dev/shm ]; then
  install -v -d -m 1777 $LFS$(realpath /dev/shm)
else
  mounttype dev/shm tmpfs tmpfs -o nosuid,nodev
fi

#mountbind usr/src
#mountbind boot
#mountbind home
EOF
```

Обратите внимание, что последние три команды в скрипте закомментированы. Они пригодятся, если эти каталоги монтируются как отдельные разделы в хост-системе и будут монтироваться при загрузке завершенной системы LFS/BLFS.

Скрипт можно запустить с помощью **bash ~/mount-virt.sh** либо от имени обычного пользователя (рекомендуется), либо от имени `root`. При запуске от имени обычного пользователя в хост-системе требуется sudo.

Еще одна проблема, на которую указывает скрипт, заключается в том, где хранить загруженные файлы пакетов. Это местоположение является произвольным. Оно может находиться в домашнем каталоге обычного пользователя, таком как ~/sources, или в глобальном каталоге /usr/src. Наша рекомендация - не смешивать источники BLFS и источники LFS в (из среды chroot) /sources. В любом случае, пакеты должны быть доступны внутри среды chroot.

Последняя удобная функция, представленная здесь, предназначена для упрощения процесса входа в среду chroot. Это можно сделать с помощью псевдонима, помещенного в пользовательский файл ~/.bashrc в хост-системе:

```
alias lfs='sudo /usr/sbin/chroot /mnt/lfs /usr/bin/env -i HOME=/root TERM="$TERM" PS1="\u:\w\\\$ "
PATH=/bin:/usr/bin:/sbin:/usr/sbin /bin/bash --login'
```

Этот псевдоним немного сложен для восприятия из-за кавычек и слэшей. Всё это должно быть в одной строке. Вышеуказанная команда была разделена на две части для презентационных целей.

11.5.2.2. Работа удаленно по ssh

Этот метод также предоставляет полноценную графическую среду, но сначала требует установки *sshd* в системе LFS, обычно в chroot. Кроме этого потребуется второй компьютер. Преимущество этого метода в том, что он прост, поскольку не требует сложной среды chroot. Он также использует собранное вами ядро LFS для всех дополнительных пакетов и по-прежнему предоставляет полную систему для установки пакетов.

Вы можете использовать команду **scp** для загрузки исходных текстов пакетов, которые будут собраны в системе LFS. Если вместо этого вы хотите загрузить исходные тексты непосредственно в систему LFS, установите *libtasn1*, *p11-kit*, *make-ca* и *wget* в chroot (или загрузите их исходники с помощью **scp** после загрузки системы LFS).

11.5.2.3. Работа из командной строки LFS

Этот метод требует установки *libtasn1*, *p11-kit*, *make-ca*, *wget*, *gpm* и *links* (или *lynx*) в chroot, а затем перезагрузки в новую систему LFS. На данный момент система по умолчанию имеет шесть виртуальных консолей. Переключать консоли так же просто, как использовать комбинации клавиш **Alt**+**Fx** , где **Fx** это клавиши от **F1** до **F6**. Комбинации **Alt**+ → и **Alt**+ ← также переключают консоль.

На этом этапе вы можете войти в две разные виртуальные консоли и запустить браузер links или lynx в одной консоли и bash в другой. GPM позволяет копировать команды из браузера с помощью левой кнопки мыши, переключать консоли и вставлять их в другую консоль.

Примечание

Вместо примечания: переключение виртуальных консолей также может быть выполнено из экземпляра X Window с помощью комбинации клавиш **Ctrl**+**Alt**+**Fx** ,но операция копирования мышью не работает между графическим интерфейсом и виртуальной консолью. Вы можете вернуться к дисплею X Window с помощью комбинации **Ctrl**+**Alt**+**Fx** ,где **Fx** обычно **F1**, но может быть **F7**.

Часть V. Приложения

Приложение A. Сокращения и условные обозначения

ABI Application Binary Interface - Двоичный (бинарный) интерфейс приложений

ALFS Automated Linux From Scratch - Проект автоматической сборки системы LFS

API Application Programming Interface - Программный интерфейс приложения

ASCII American Standard Code for Information Interchange — Американский стандартный код для обмена информацией

BIOS Basic Input/Output System - Базовая система ввода/вывода

BLFS Beyond Linux From Scratch - Проект, расширяющий возможности Linux From Scratch

BSD Berkeley Software Distribution - Система распространения программного обеспечения в исходных кодах

chroot change root - Команда изменения корневого каталога

CMOS Complementary Metal Oxide Semiconductor - Комплементарная структура металл-оксид-полупроводник

COS Class Of Service - Класс обслуживания

CPU Central Processing Unit - Центральный процессор, процессор

CRC Cyclic Redundancy Check - Циклический избыточный код

CVS Concurrent Versions System - Централизованная система управления версиями

DHCP Dynamic Host Configuration Protocol - Протокол динамической настройки узла

DNS Domain Name Service - Служба доменных имён

EGA Enhanced Graphics Adapter - Усовершенствованный графический адаптер

ELF Executable and Linkable Format - Формат исполняемых и компонуемых файлов

EOF End of File - Конец файла, символ конца файла

EQN equation - уравнение

ext2 second extended file system - вторая расширенная файловая система

ext3 third extended file system - третья расширенная файловая система

ext4 fourth extended file system - четвёртая расширенная файловая система

FAQ Frequently Asked Questions - Часто задаваемые вопросы

FHS Filesystem Hierarchy Standard - Стандарт иерархии файловой системы

FIFO First-In, First Out - Схема обслуживания очереди "первый пришел — первым ушёл"

FQDN Fully Qualified Domain Name - Полное доменное имя

FTP File Transfer Protocol - Протокол передачи файлов

GB Gigabytes - Гигабайты

GCC GNU Compiler Collection - Коллекция компиляторов GNU

GID Group Identifier - Идентификатор группы

GMT Greenwich Mean Time - Среднее время по Гринвичу

HTML Hypertext Markup Language - Язык гипертекстовой разметки

IDE Integrated Drive Electronics - Интерфейс подключения дисковых устройств

IEEE Institute of Electrical and Electronic Engineers - Институт инженеров электротехники и электроники

IO Input/Output - Ввод/вывод

IP Internet Protocol - Межсетевой протокол

IPC Inter-Process Communication - Обмен данными между потоками одного или разных процессов

IRC Internet Relay Chat - Ретранслируемый интернет-чат

ISO International Organization for Standardization - Международная организация по стандартизации

ISP Internet Service Provider - Провайдер интернет услуг

KB Kilobytes - Килобайты

LED Light Emitting Diode - Светодиод

LFS Linux From Scratch - Линукс с нуля

LSB Linux Standard Base - Совместный проект семейства операционных систем, основанных на Linux (то есть дистрибутивов Linux), при организации Linux Foundation, целью которого является стандартизация их внутренней структуры. LSB опирается на существующие спецификации, такие как POSIX, Single UNIX Specification, и другие открытые стандарты, при этом расширяя и дополняя их.

MB Megabytes - Мегабайты

MBR Master Boot Record - Главная загрузочная запись

MD5 Message Digest 5 - 128-битный алгоритм хеширования

NIC Network Interface Card - Сетевой адаптер

NLS Native Language Support - Поддержка естественного языка

NNTP Network News Transport Protocol - Сетевой транспортный протокол новостных групп

NPTL Native POSIX Threading Library - Библиотека потоков POSIX

OSS Open Sound System - Унифицированный драйвер для звуковых карт и других звуковых устройств

PCH Pre-Compiled Headers - Предварительно скомпилированные заголовки

PCRE Perl Compatible Regular Expression - Регулярные выражения, совместимые с Perl

PID Process Identifier - Идентификатор процесса

PTY pseudo terminal - Псевдотерминал

QOS Quality Of Service - Качество обслуживания

RAM Random Access Memory - Оперативная память

RPC Remote Procedure Call - Удаленный вызов процедур

RTC Real Time Clock - Часы реального времени

SBU Standard Build Unit - Стандартная единица (времени) сборки

SCO The Santa Cruz Operation - Компания-разработчик программного обеспечения

SHA1 Secure-Hash Algorithm 1 - Алгоритм криптографического хеширования

TLDP The Linux Documentation Project - Проект документации Linux

TFTP Trivial File Transfer Protocol - Простейший протокол передачи файлов

TLS Thread-Local Storage - Локальное хранилище потока

UID User Identifier - Идентификатор пользователя

umask user file-creation mask - Команда, определяющая маску создания пользовательских файлов

USB Universal Serial Bus - Универсальная последовательная шина

UTC Coordinated Universal Time - Всемирное координированное время

UUID Universally Unique Identifier - Универсальный уникальный идентификатор

VC Virtual Console - Виртуальная консоль

VGA Video Graphics Array - Компонентный видеоинтерфейс

VT Virtual Terminal - Виртуальный терминал

Приложение B. Благодарности

Мы хотели бы поблагодарить следующих людей и организации за их вклад в проект Linux From Scratch.

- *Gerard Beekmans* <gerard@linuxfromscratch.org> – Основатель проекта LFS
- *Bruce Dubbs* <bdubbs@linuxfromscratch.org> – Главный редактор LFS
- *Jim Gifford* <jim@linuxfromscratch.org> – Второй руководитель проекта CLFS
- *Pierre Labastie* <pierre@linuxfromscratch.org> – Редактор BLFS и руководитель ALFS
- *DJ Lucas* <dj@linuxfromscratch.org> – Редактор проектов LFS и BLFS
- *Ken Moffat* <ken@linuxfromscratch.org> – Редактор BLFS
- Бесчисленное множество других людей из различных списков рассылки проектов LFS и BLFS, которые помогали в создании этой книги, присылая свои предложения, проверяя книгу и отправляя отчеты об ошибках, инструкции и собственный опыт установки различных пакетов.

Переводчики

- *Manuel Canales Esparcia* <macana@macana-es.com> – Перевод проекта LFS на испанский язык
- *Johan Lenglet* <johan@linuxfromscratch.org> – Перевод проекта LFS на французский язык до 2008 г.
- *Jean-Philippe Mengual* <jmengual@linuxfromscratch.org> – Перевод проекта LFS на французский язык 2008-2016 гг
- *Julien Lepiller* <jlepiller@linuxfromscratch.org> – Перевод проекта LFS на французский язык с 2017-по настоящее время
- *Anderson Lizardo* <lizardo@linuxfromscratch.org> – Перевод проекта LFS на португальский язык до 2022 г.
- *Jamenson Espindula* <jafesp@gmail.com> – Перевод проекта LFS на португальский язык 2022-по настоящее время
- *Thomas Reitelbach* <tr@erdfunkstelle.de> – Перевод проекта LFS на немецкий язык
- *Anton Maisak* <info@linuxfromscratch.ru> – Перевод проекта LFS на русский язык 2018-2020 гг
- *Elena Shevcova* <info@linuxfromscratch.ru> – Перевод проекта LFS на русский язык 2018-2020 гг
- *Vladimir Pertsev* <info@linuxfromscratch.ru> – Перевод проекта LFS на русский язык 2022-по настоящее время

Зеркала проекта

Североамериканские зеркала

- *Scott Kveton* <scott@osuosl.org> – зеркало lfs.oregonstate.edu
- *William Astle* <lost@l-w.net> – зеркало ca.linuxfromscratch.org
- *Eujon Sellers* <jpolen@rackspace.com> – зеркало lfs.introspeed.com
- *Justin Knierim* <tim@idge.net> – зеркало lfs-matrix.net

Южноамериканские зеркала

- *Manuel Canales Esparcia* <manuel@linuxfromscratch.org> – зеркало lfsmirror.lfs-es.info
- *Luis Falcon* <Luis Falcon> – зеркало torredehanoi.org

Европейские зеркала

- *Guido Passet* <guido@primerelay.net> – зеркало nl.linuxfromscratch.org
- *Bastiaan Jacques* <baafie@planet.nl> – зеркало lfs.pagefault.net
- *Sven Cranshoff* <sven.cranshoff@lineo.be> – зеркало lfs.lineo.be
- Scarlet Belgium – зеркало lfs.scarlet.be
- *Sebastian Faulborn* <info@aliensoft.org> – зеркало lfs.aliensoft.org
- *Stuart Fox* <stuart@dontuse.ms> – зеркало lfs.dontuse.ms
- *Ralf Uhlemann* <admin@realhost.de> – зеркало lfs.oss-mirror.org
- *Antonin Sprinzl* <Antonin.Sprinzl@tuwien.ac.at> – зеркало at.linuxfromscratch.org
- *Fredrik Danerklint* <fredan-lfs@fredan.org> – зеркало se.linuxfromscratch.org
- *Franck* <franck@linuxpourtous.com> – зеркало lfs.linuxpourtous.com
- *Philippe Baque* <baque@cict.fr> – зеркало lfs.cict.fr
- *Benjamin Heil* <kontakt@wankoo.org> – зеркало lfs.wankoo.org
- *Vladimir Pertsev* <info@linuxfromscratch.ru> – зеркало mirror.linuxfromscratch.ru

Азиатские зеркала

- *Satit Phermsawang* <satit@wbac.ac.th> – зеркало lfs.phayoune.org
- *Shizunet Co.,Ltd.* <info@shizu-net.jp> – зеркало lfs.mirror.shizu-net.jp

Австралийские зеркала

- *Jason Andrade* <jason@dstc.edu.au> – зеркало au.linuxfromscratch.org

Бывшие участники проекта

- *Christine Barczak* <theladyskye@linuxfromscratch.org> – Редактор книги LFS
- Archaic <archaic@linuxfromscratch.org> – Технический писатель/редактор LFS, руководитель проекта HLFS, редактор BLFS, Сопровождающий проекта Советы и патчи
- *Matthew Burgess* <matthew@linuxfromscratch.org> – Руководитель проекта LFS, технический писатель/редактор LFS
- *Nathan Coulson* <nathan@linuxfromscratch.org> – Сопровождающий LFS-Bootscripts
- Timothy Bauscher
- Robert Briggs
- Ian Chilton
- *Jeroen Coumans* <jeroen@linuxfromscratch.org> – Разработчик веб-сайта, сопровождающий FAQ
- *Manuel Canales Esparcia* <manuel@linuxfromscratch.org> – Сопровождающий XML и XSL проектов LFS/BLFS/HLFS
- Alex Groenewoud – Технический писатель LFS
- Marc Heerdink
- *Jeremy Huntwork* <jhuntwork@linuxfromscratch.org> – Технический писатель LFS, сопровождающий LFS LiveCD
- *Bryan Kadzban* <bryan@linuxfromscratch.org> – Технический писатель LFS

- Mark Hymers
- Seth W. Klein – Сопровождающий FAQ
- *Nicholas Leippe* <nicholas@linuxfromscratch.org> – Сопровождающий Wiki
- *Anderson Lizardo* <lizardo@linuxfromscratch.org> – Сопровождающий движка сайта
- *Randy McMurchy* <randy@linuxfromscratch.org> – Руководитель проекта BLFS, редактор LFS
- *Dan Nicholson* <dnicholson@linuxfromscratch.org> – Редактор LFS и BLFS
- *Alexander E. Patrakov* <alexander@linuxfromscratch.org> – Технический писатель LFS, редактор интернационализации LFS, сопровождающий LFS Live CD
- Simon Perreault
- *Scot Mc Pherson* <scot@linuxfromscratch.org> – Сопровождающий шлюза NNTP для проекта LFS
- *Douglas R. Reno* <renodr@linuxfromscratch.org> – Редактор Systemd
- *Ryan Oliver* <ryan@linuxfromscratch.org> – Один из руководителей проекта CLFS
- *Greg Schafer* <gschafer@zip.com.au> – Технический писатель проекта LFS и архитектор методов сборки пакетов следующего поколения, предназначенных для 64-битной архитектуры
- Jesse Tie-Ten-Quee – Технический писатель LFS
- *James Robertson* <jwrober@linuxfromscratch.org> – Сопровождающий Bugzilla
- *Tushar Teredesai* <tushar@linuxfromscratch.org> – Редактор книги BLFS, руководитель проекта Советы и Патчи
- *Jeremy Utley* <jeremy@linuxfromscratch.org> – Технический писатель проекта LFS, сопровождающий Bugzilla, сопровождающий LFS-Bootscripts
- *Zack Winkles* <zwinkles@gmail.com> – Технический писатель проекта LFS

Приложение C. Зависимости

Каждый пакет в системе LFS для правильной сборки и установки может ссылаться на один или несколько других пакетов. Некоторые пакеты могут иметь циклические зависимости, то есть первый пакет зависит от второго, который в свою очередь, зависит от первого. Именно по этой причине, указанный порядок сборки пакетов в LFS очень важен. Цель этой страницы - документировать зависимости каждого пакета, собранного в LFS

Для каждого собираемого пакета существует от трёх до пяти типов зависимостей, перечисленных ниже. В первом списке перечислены другие пакеты, которые должны быть доступны для компиляции и установки рассматриваемого пакета. Во втором перечислены пакеты, которые должны быть доступны, когда какие-либо программы или библиотеки из пакета используются во время его выполнения. В третьем списке перечислены пакеты, которые, в дополнение к пакетам из первого списка, должны быть доступны для запуска наборов тестов. Четвертый список зависимостей — это пакеты, которые требуют, чтобы некий пакет был собран и установлен по определенному пути, прежде чем они будут собраны и установлены.

Последний список зависимостей - это необязательные пакеты, которые не рассматриваются в LFS, но могут быть полезны пользователю. Эти пакеты могут иметь дополнительные как обязательные, так и необязательные зависимости. Такие зависимости - рекомендуется разрешать после завершения сборки всей системы LFS. В некоторых случаях, повторная установка некоторых таких пакетов рассматривается в BLFS.

Acl

Установка зависит от: Attr, Bash, Binutils, Coreutils, GCC, Gettext, Grep, M4, Make, Perl, Sed и Texinfo

Требуется во время выполнения: Attr и Glibc

Набор тестов зависит от: Automake, Diffutils, Findutils и Libtool

Должен быть установлен до: Coreutils, Sed, Tar и Vim

Необязательные зависимости: Нет

Attr

Установка зависит от: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, M4, Make, Perl, Sed и Texinfo

Требуется во время выполнения: Glibc

Набор тестов зависит от: Automake, Diffutils, Findutils и Libtool

Должен быть установлен до: Acl, Libcap и Patch

Необязательные зависимости: Нет

Autoconf

Установка зависит от: Bash, Coreutils, Grep, M4, Make, Perl, Sed и Texinfo

Требуется во время выполнения: Bash, Coreutils, Grep, M4, Make, Sed и Texinfo

Набор тестов зависит от: Automake, Diffutils, Findutils, GCC и Libtool

Должен быть установлен до: Automake и Coreutils

Необязательные зависимости: *Emacs*

Automake

Установка зависит от: Autoconf, Bash, Coreutils, Gettext, Grep, M4, Make, Perl, Sed, и Texinfo

Требуется во время выполнения: Bash, Coreutils, Grep, M4, Sed и Texinfo

Набор тестов зависит от: Binutils, Bison, Bzip2, DejaGNU, Diffutils, Expect, Findutils, Flex, GCC, Gettext, Gzip, Libtool и Tar

Должен быть установлен до: Coreutils

Необязательные зависимости: Нет

Bash

Установка зависит от: Bash, Binutils, Bison, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Make, Ncurses, Patch, Readline, Sed и Texinfo

Требуется во время выполнения: Glibc, Ncurses и Readline

Набор тестов зависит от: Expect и Shadow

Должен быть установлен до: Нет

Необязательные зависимости: *Xorg*

Bc

Установка зависит от: Bash, Binutils, Coreutils, GCC, Glibc, Grep, Make и Readline

Требуется во время выполнения: Glibc, Ncurses и Readline

Набор тестов зависит от: Gawk

Должен быть установлен до: Linux

Необязательные зависимости: Нет

Binutils

Установка зависит от: Bash, Binutils, Coreutils, Diffutils, File, Flex, Gawk, GCC, Glibc, Grep, Make, Perl, Pkgconf, Sed, Texinfo, Zlib и Zstd

Требуется во время выполнения: Glibc, Zlib и Zstd

Набор тестов зависит от: DejaGNU и Expect

Должен быть установлен до: Нет

Необязательные зависимости: *Elfutils* и *Jansson*

Bison

Установка зависит от: Bash, Binutils, Coreutils, Diffutils, GCC, Gettext, Glibc, Grep, M4, Make, Perl и Sed

Требуется во время выполнения: Glibc

Набор тестов зависит от: Diffutils, Findutils и Flex

Должен быть установлен до: Kbd и Tar

Необязательные зависимости: *Doxygen*

Bzip2

Установка зависит от: Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Make и Patch

Требуется во время выполнения: Glibc

Набор тестов зависит от: Нет

Должен быть установлен до: File и Libelf

Необязательные зависимости: Нет

Check

Установка зависит от: Gawk, GCC, Grep, Make, Sed и Texinfo

Требуется во время выполнения: Bash и Gawk

Набор тестов зависит от: Нет

Должен быть установлен до: Нет

Необязательные зависимости: *libsubunit* и *patchutils*

Coreutils

Установка зависит от: Autoconf, Automake, Bash, Binutils, Coreutils, GCC, Gettext, Glibc, GMP, Grep, Libcap, Make, OpenSSL, Patch, Perl, Sed и Texinfo

Требуется во время выполнения: Glibc

Набор тестов зависит от: Diffutils, E2fsprogs, Findutils, Shadow и Util-linux

Должен быть установлен до: Bash, Diffutils, Findutils, Man-DB и Udev

Необязательные зависимости: *Expect.pm* и *IO::Tty*

DejaGNU

Установка зависит от: Bash, Coreutils, Diffutils, Expect, GCC, Grep, Make, Sed и Texinfo

Требуется во время выполнения: Expect и Bash

Набор тестов зависит от: Нет

Должен быть установлен до: Нет

Необязательные зависимости: Нет

Diffutils

Установка зависит от: Bash, Binutils, Coreutils, Gawk, GCC, Gettext, Glibc, Grep, Make, Sed и Texinfo

Требуется во время выполнения: Glibc

Набор тестов зависит от: Perl

Должен быть установлен до: Нет

Необязательные зависимости: Нет

E2fsprogs

Установка зависит от: Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Gzip, Make, Pkgconf, Sed, Texinfo и Util-linux

Требуется во время выполнения: Glibc и Util-linux

Набор тестов зависит от: Procps-ng и Psmisc

Должен быть установлен до: Нет

Необязательные зависимости: Нет

Expat

Установка зависит от: Bash, Binutils, Coreutils, Gawk, GCC, Glibc, Grep, Make и Sed

Требуется во время выполнения: Glibc

Набор тестов зависит от: Нет

Должен быть установлен до: Python и XML::Parser

Необязательные зависимости: Нет

Expect

Установка зависит от: Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Grep, Make, Patch, Sed и Tcl

Требуется во время выполнения: Glibc и Tcl

Набор тестов зависит от: Нет

Должен быть установлен до: Нет

Необязательные зависимости: *Tk*

File

Установка зависит от: Bash, Binutils, Bzip2, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Make, Sed, Xz и Zlib

Требуется во время выполнения: Glibc, Bzip2, Xz и Zlib

Набор тестов зависит от: Нет

Должен быть установлен до: Нет

Необязательные зависимости: *libseccomp*

Findutils

Установка зависит от: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, Make, Sed, и Texinfo

Требуется во время выполнения: Bash и Glibc

Набор тестов зависит от: DejaGNU, Diffutils и Expect

Должен быть установлен до: Нет

Необязательные зависимости: Нет

Flex

Установка зависит от: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, M4, Make, Patch, Sed и Texinfo

Требуется во время выполнения: Bash, Glibc и M4

Набор тестов зависит от: Bison и Gawk

Должен быть установлен до: Binutils, IProute2, Kbd, Kmod и Man-DB

Необязательные зависимости: Нет

Flit-Core

Установка зависит от: Python

Требуется во время выполнения: Python

Набор тестов зависит от: Набор тестов недоступен

Должен быть установлен до: Wheel

Необязательные зависимости: *pytest* и *testpath*

Gawk

Установка зависит от: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, GMP, Grep, Make, MPFR, Patch, Readline, Sed и Texinfo

Требуется во время выполнения: Bash, Glibc и Mpfr

Набор тестов зависит от: Diffutils

Должен быть установлен до: Нет

Необязательные зависимости: *libsigsegv*

GCC

Установка зависит от: Bash, Binutils, Coreutils, Diffutils, Findutils, Gawk, GCC, Gettext, Glibc, GMP, Grep, M4, Make, MPC, MPFR, Patch, Perl, Sed, Tar, Texinfo и Zstd

Требуется во время выполнения: Bash, Binutils, Glibc, Mpc и Python

Набор тестов зависит от: DejaGNU, Expect и Shadow

Должен быть установлен до: Нет

Необязательные зависимости: *GDC*, *GNAT*, и *ISL*

GDBM

Установка зависит от: Bash, Binutils, Coreutils, Diffutils, GCC, Grep, Make и Sed

Требуется во время выполнения: Bash, Glibc и Readline

Набор тестов зависит от: Нет

Должен быть установлен до: Нет

Необязательные зависимости: Нет

Gettext

Установка зависит от: Bash, Binutils, Coreutils, Gawk, GCC, Glibc, Grep, Make, Ncurses, Sed и Texinfo

Требуется во время выполнения: Acl, Bash, Gcc и Glibc

Набор тестов зависит от: Diffutils, Perl и Tcl

Должен быть установлен до: Automake и Bison

Необязательные зависимости: *libunistring* и *libxml2*

Glibc

Установка зависит от: Bash, Binutils, Bison, Coreutils, Diffutils, Gawk, GCC, Gettext, Grep, Gzip, Linux API Headers, Make, Perl, Python, Sed и Texinfo

Требуется во время выполнения: Нет

Набор тестов зависит от: File

Должен быть установлен до: Нет

Необязательные зависимости: Нет

GMP

Установка зависит от: Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, M4, Make, Sed и Texinfo

Требуется во время выполнения: GCC и Glibc

Набор тестов зависит от: Нет

Должен быть установлен до: MPFR и GCC

Необязательные зависимости: Нет

Gperf

Установка зависит от: Bash, Binutils, Coreutils, GCC, Glibc и Make

Требуется во время выполнения: GCC и Glibc

Набор тестов зависит от: Diffutils и Expect

Должен быть установлен до: Нет

Необязательные зависимости: Нет

Grep

Установка зависит от: Bash, Binutils, Coreutils, Diffutils, GCC, Gettext, Glibc, Grep, Make, Patch, Sed и Texinfo

Требуется во время выполнения: Glibc

Набор тестов зависит от: Gawk

Должен быть установлен до: Man-DB

Необязательные зависимости: *PCRE2* и *libsigsegv*

Groff

Установка зависит от: Bash, Binutils, Bison, Coreutils, Gawk, GCC, Glibc, Grep, Make, Patch, Sed и Texinfo

Требуется во время выполнения: GCC, Glibc и Perl

Набор тестов зависит от: Набор тестов недоступен

Должен быть установлен до: Man-DB

Необязательные зависимости: *ghostscript* и *Uchardet*

GRUB

Установка зависит от: Bash, Binutils, Bison, Coreutils, Diffutils, GCC, Gettext, Glibc, Grep, Make, Ncurses, Sed, Texinfo и Xz

Требуется во время выполнения: Bash, GCC, Gettext, Glibc, Xz и Sed.

Набор тестов зависит от: Нет

Должен быть установлен до: Нет

Необязательные зависимости: Нет

Gzip

Установка зависит от: Bash, Binutils, Coreutils, GCC, Glibc, Grep, Make, Sed и Texinfo

Требуется во время выполнения: Bash и Glibc

Набор тестов зависит от: Diffutils и Less

Должен быть установлен до: Man-DB

Необязательные зависимости: Нет

Iana-Etc

Установка зависит от: Coreutils

Требуется во время выполнения: Нет

Набор тестов зависит от: Набор тестов недоступен

Должен быть установлен до: Perl

Необязательные зависимости: Нет

Inetutils

Установка зависит от: Bash, Binutils, Coreutils, GCC, Glibc, Grep, Make, Ncurses, Patch, Sed, Texinfo и Zlib

Требуется во время выполнения: GCC, Glibc, Ncurses и Readline

Набор тестов зависит от: Нет

Должен быть установлен до: Tar

Необязательные зависимости: Нет

Intltool

Установка зависит от: Bash, Gawk, Glibc, Make, Perl, Sed и XML::Parser

Требуется во время выполнения: Autoconf, Automake, Bash, Glibc, Grep, Perl и Sed

Набор тестов зависит от: Perl

Должен быть установлен до: Нет

Необязательные зависимости: Нет

IProute2

Установка зависит от: Bash, Bison, Coreutils, Flex, GCC, Glibc, Make, Libcap, Libelf, Linux API Headers, Pkgconf и Zlib

Требуется во время выполнения: Bash, Coreutils, Glibc, Libcap, Libelf и Zlib

Набор тестов зависит от: Набор тестов недоступен

Должен быть установлен до: Нет

Необязательные зависимости: *Berkeley DB*, *iptables*, *libbpf*, *libmnl* и *libtirpc*

Jinja2

Установка зависит от: MarkupSafe, Python, Setuptools и Wheel

Требуется во время выполнения: MarkupSafe и Python

Набор тестов зависит от: Набор тестов недоступен

Должен быть установлен до: Udev

Необязательные зависимости: Нет

Kbd

Установка зависит от: Bash, Binutils, Bison, Check, Coreutils, Flex, GCC, Gettext, Glibc, Gzip, Make, Patch и Sed

Требуется во время выполнения: Bash, Coreutils и Glibc

Набор тестов зависит от: Нет

Должен быть установлен до: Нет

Необязательные зависимости: *Linux-PAM*

Kmod

Установка зависит от: Bash, Binutils, Bison, Coreutils, Flex, GCC, Gettext, Glibc, Gzip, Make, OpenSSL, Pkgconf, Sed, Xz и Zlib

Требуется во время выполнения: Glibc, Xz и Zlib

Набор тестов зависит от: Набор тестов недоступен

Должен быть установлен до: Udev

Необязательные зависимости: *scdoc* (для справочных страниц)

Less

Установка зависит от: Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Grep, Make, Ncurses и Sed

Требуется во время выполнения: Glibc и Ncurses

Набор тестов зависит от: Набор тестов недоступен

Должен быть установлен до: Gzip

Необязательные зависимости: *PCRE2* или *PCRE*

Libcap

Установка зависит от: Attr, Bash, Binutils, Coreutils, GCC, Glibc, Perl, Make и Sed

Требуется во время выполнения: Glibc

Набор тестов зависит от: Нет

Должен быть установлен до: IProute2 и Shadow

Необязательные зависимости: *Linux-PAM*

Libelf

Установка зависит от: Bash, Binutils, Bzip2, Coreutils, GCC, Glibc, Make, Xz, Zlib и Zstd

Требуется во время выполнения: Bzip2, Glibc, Xz, Zlib и Zstd

Набор тестов зависит от: Нет

Должен быть установлен до: IProute2 и Linux

Необязательные зависимости: Нет

Libffi

Установка зависит от: Bash, Binutils, Coreutils, GCC, Glibc, Make и Sed

Требуется во время выполнения: Glibc

Набор тестов зависит от: DejaGnu

Должен быть установлен до: Python

Необязательные зависимости: Нет

Libpipeline

Установка зависит от: Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Make, Sed и Texinfo

Требуется во время выполнения: Glibc

Набор тестов зависит от: Check и Pkgconf

Должен быть установлен до: Man-DB

Необязательные зависимости: Нет

Libtool

Установка зависит от: Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Make, Sed и Texinfo

Требуется во время выполнения: Autoconf, Automake, Bash, Binutils, Coreutils, File, GCC, Glibc, Grep, Make и Sed

Набор тестов зависит от: Autoconf, Automake и Findutils

Должен быть установлен до: Нет

Необязательные зависимости: Нет

Libxcrypt

Установка зависит от: Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Make, Perl и Sed

Требуется во время выполнения: Glibc

Набор тестов зависит от: Нет

Должен быть установлен до: Perl, Python, Shadow и Udev

Необязательные зависимости: Нет

Linux

Установка зависит от: Bash, Bc, Binutils, Coreutils, Diffutils, Findutils, GCC, Glibc, Grep, Gzip, Kmod, Libelf, Make, Ncurses, OpenSSL, Perl и Sed

Требуется во время выполнения: Нет

Набор тестов зависит от: Набор тестов недоступен

Должен быть установлен до: Нет

Необязательные зависимости: *cpio* и *LLVM* (с Clang) и *Rust-bindgen*

Linux API Headers

Установка зависит от: Bash, Binutils, Coreutils, Findutils, GCC, Glibc, Grep, Gzip, Make, Perl и Sed

Требуется во время выполнения: Нет

Набор тестов зависит от: Набор тестов недоступен

Должен быть установлен до: Нет

Необязательные зависимости: Нет

Lz4

Установка зависит от: Bash, Binutils, Coreutils, GCC, Glibc и Make

Требуется во время выполнения: Glibc

Набор тестов зависит от: Python

Должен быть установлен до: Zstd

Необязательные зависимости: Нет

M4

Установка зависит от: Bash, Binutils, Coreutils, GCC, Glibc, Grep, Make, Sed и Texinfo

Требуется во время выполнения: Bash и Glibc

Набор тестов зависит от: Diffutils

Должен быть установлен до: Autoconf и Bison

Необязательные зависимости: *libsigsegv*

Make

Установка зависит от: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, Make, Sed и Texinfo

Требуется во время выполнения: Glibc

Набор тестов зависит от: Perl и Procps-ng

Должен быть установлен до: Нет

Необязательные зависимости: *Guile*

Man-DB

Установка зависит от: Bash, Binutils, Bzip2, Coreutils, Flex, GCC, GDBM, Gettext, Glibc, Grep, Groff, Gzip, Less, Libpipeline, Make, Pkgconf, Sed, и Xz

Требуется во время выполнения: Bash, GDBM, Groff, Glibc, Gzip, Less, Libpipeline и Zlib

Набор тестов зависит от: Util-linux

Должен быть установлен до: Нет

Необязательные зависимости: *libseccomp* и *po4a*

Man-Pages

Установка зависит от: Bash, Coreutils, Make и Sed

Требуется во время выполнения: Нет

Набор тестов зависит от: Набор тестов недоступен

Должен быть установлен до: Нет

Необязательные зависимости: Нет

MarkupSafe

Установка зависит от: Python, Setuptools и Wheel

Требуется во время выполнения: Python

Набор тестов зависит от: Набор тестов недоступен

Должен быть установлен до: Jinja2

Необязательные зависимости: Нет

Meson

Установка зависит от: Ninja, Python, Setuptools и Wheel

Требуется во время выполнения: Python

Набор тестов зависит от: Набор тестов недоступен

Должен быть установлен до: Udev

Необязательные зависимости: Нет

MPC

Установка зависит от: Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, GMP, Make, MPFR, Sed и Texinfo

Требуется во время выполнения: Glibc, GMP и MPFR

Набор тестов зависит от: Нет

Должен быть установлен до: GCC

Необязательные зависимости: Нет

MPFR

Установка зависит от: Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, GMP, Make, Sed и Texinfo

Требуется во время выполнения: Glibc и GMP

Набор тестов зависит от: Нет

Должен быть установлен до: Gawk и GCC

Необязательные зависимости: Нет

Ncurses

Установка зависит от: Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Make, Patch и Sed

Требуется во время выполнения: Glibc

Набор тестов зависит от: Набор тестов недоступен

Должен быть установлен до: Bash, GRUB, Inetutils, Less, Procps-ng, Psmisc, Readline, Texinfo, Util-linux и Vim

Необязательные зависимости: Нет

Ninja

Установка зависит от: Binutils, Coreutils, GCC и Python

Требуется во время выполнения: GCC и Glibc

Набор тестов зависит от: *cmake*

Должен быть установлен до: Meson

Необязательные зависимости: *Asciidoc*, *Doxygen*, *Emacs* и *re2c*

OpenSSL

Установка зависит от: Binutils, Coreutils, GCC, Make и Perl

Требуется во время выполнения: Glibc и Perl

Набор тестов зависит от: Нет

Должен быть установлен до: Coreutils, Kmod, Linux и Udev

Необязательные зависимости: Нет

Patch

Установка зависит от: Attr, Bash, Binutils, Coreutils, GCC, Glibc, Grep, Make и Sed

Требуется во время выполнения: Attr и Glibc

Набор тестов зависит от: Diffutils

Должен быть установлен до: Нет

Необязательные зависимости: *Ed*

Perl

Установка зависит от: Bash, Binutils, Coreutils, Gawk, GCC, GDBM, Glibc, Grep, Libxcrypt, Make, Sed и Zlib

Требуется во время выполнения: GDBM, Glibc и Libxcrypt

Набор тестов зависит от: Iana-Etc, Less и Procps-ng

Должен быть установлен до: Autoconf

Необязательные зависимости: *Berkeley DB*

Pkgconf

Установка зависит от: Bash, Binutils, Coreutils, Gawk, GCC, Glibc, Grep, Make и Sed

Требуется во время выполнения: Glibc

Набор тестов зависит от: Нет

Должен быть установлен до: Binutils, E2fsprogs, IProute2, Kmod, Man-DB, Procps-ng, Python, Udev и Util-linux

Необязательные зависимости: Нет

Procps-ng

Установка зависит от: Bash, Binutils, Coreutils, GCC, Glibc, Make, Ncurses, и Pkgconf

Требуется во время выполнения: Glibc

Набор тестов зависит от: DejaGNU

Должен быть установлен до: Нет

Необязательные зависимости: *elogind*

Psmisc

Установка зависит от: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, Make, Ncurses и Sed

Требуется во время выполнения: Glibc и Ncurses

Набор тестов зависит от: Набор тестов недоступен

Должен быть установлен до: Нет

Необязательные зависимости: Нет

Python

Установка зависит от: Bash, Binutils, Coreutils, Expat, GCC, Gdbm, Gettext, Glibc, Grep, Libffi, Libxcrypt, Make, Ncurses, OpenSSL, Pkgconf, Sed и Util-linux

Требуется во время выполнения: Bzip2, Expat, Gdbm, Glibc, Libffi, Libxcrypt, Ncurses, OpenSSL и Zlib

Набор тестов зависит от: GDB и Valgrind

Должен быть установлен до: Ninja

Необязательные зависимости: *Berkeley DB*, *libnsl*, *SQLite* и *Tk*

Readline

Установка зависит от: Bash, Binutils, Coreutils, Gawk, GCC, Glibc, Grep, Make, Ncurses, Patch, Sed и Texinfo

Требуется во время выполнения: Glibc и Ncurses

Набор тестов зависит от: Набор тестов недоступен

Должен быть установлен до: Bash, Bc и Gawk

Необязательные зависимости: Нет

Sed

Установка зависит от: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, Make, Sed, и Texinfo

Требуется во время выполнения: Acl, Attr и Glibc

Набор тестов зависит от: Diffutils и Gawk

Должен быть установлен до: E2fsprogs, File, Libtool и Shadow

Необязательные зависимости: Нет

Setuptools

Установка зависит от: Python и Wheel

Требуется во время выполнения: Python

Набор тестов зависит от: Набор тестов недоступен

Должен быть установлен до: Jinja2, MarkupSafe и Meson

Необязательные зависимости: Нет

Shadow

Установка зависит от: Acl, Attr, Bash, Binutils, Coreutils, Diffutils, Findutils, Gawk, GCC, Gettext, Glibc, Grep, Libcap, Libxcrypt, Make и Sed

Требуется во время выполнения: Glibc и Libxcrypt

Набор тестов зависит от: Набор тестов недоступен

Должен быть установлен до: Coreutils

Необязательные зависимости: *CrackLib* и *Linux-PAM*

Sysklogd

Установка зависит от: Binutils, Coreutils, GCC, Glibc, Make и Patch

Требуется во время выполнения: Glibc

Набор тестов зависит от: Набор тестов недоступен

Должен быть установлен до: Нет

Необязательные зависимости: Нет

SysVinit

Установка зависит от: Binutils, Coreutils, GCC, Glibc, Make и Sed

Требуется во время выполнения: Glibc

Набор тестов зависит от: Набор тестов недоступен

Должен быть установлен до: Нет

Необязательные зависимости: Нет

Tar

Установка зависит от: Acl, Attr, Bash, Binutils, Bison, Coreutils, GCC, Gettext, Glibc, Grep, Inetutils, Make, Sed и Texinfo

Требуется во время выполнения: Acl, Attr, Bzip2, Glibc, Gzip и Xz

Набор тестов зависит от: Autoconf, Diffutils, Findutils, Gawk и Gzip

Должен быть установлен до: Нет

Необязательные зависимости: Нет

Tcl

Установка зависит от: Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Grep, Make и Sed

Требуется во время выполнения: Glibc и Zlib

Набор тестов зависит от: Нет

Должен быть установлен до: Нет

Необязательные зависимости: Нет

Texinfo

Установка зависит от: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, Make, Ncurses, Patch и Sed

Требуется во время выполнения: Glibc и Ncurses

Набор тестов зависит от: Нет

Должен быть установлен до: Нет

Необязательные зависимости: Нет

Udev

Установка зависит от: Acl, Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Gperf, Grep, Jinja2, Libcap, Libxcrypt, Meson, OpenSSL, Pkgconf, Sed, Util-linux и Zstd

Требуется во время выполнения: Acl, Glibc, Libcap, OpenSSL и Util-linux

Набор тестов зависит от: Нет

Должен быть установлен до: Util-linux

Необязательные зависимости: Нет

Util-linux

Установка зависит от: Bash, Binutils, Coreutils, Diffutils, File, Findutils, Gawk, GCC, Gettext, Glibc, Grep, Make, Ncurses, Pkgconf, Sed, Udev и Zlib

Требуется во время выполнения: Glibc, Ncurses, Readline, Udev и Zlib

Набор тестов зависит от: Нет

Должен быть установлен до: Нет

Необязательные зависимости: *Asciidoctor*, *Libcap-NG*, *libeconf*, *libuser*, *libutempter*, *Linux-PAM*, *smartmontools*, *po4a* и *slang*

Vim

Установка зависит от: Acl, Attr, Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Grep, Make, Ncurses и Sed

Требуется во время выполнения: Acl, Attr, Glibc, Python, Ncurses и Tcl

Набор тестов зависит от: Нет

Должен быть установлен до: Нет

Необязательные зависимости: *Xorg*, *GTK+ 2*, *LessTif*, *Ruby* и *GPM*

Wheel

Установка зависит от: Python и Flit-core

Требуется во время выполнения: Python

Набор тестов зависит от: Набор тестов недоступен

Должен быть установлен до: Jinja2, MarkupSafe, Meson и Setuptools

Необязательные зависимости: Нет

XML::Parser

Установка зависит от: Bash, Binutils, Coreutils, Expat, GCC, Glibc, Make и Perl
Требуется во время выполнения: Expat, Glibc и Perl
Набор тестов зависит от: Perl
Должен быть установлен до: Intltool
Необязательные зависимости: Нет

Xz

Установка зависит от: Bash, Binutils, Coreutils, Diffutils, GCC, Glibc и Make
Требуется во время выполнения: Glibc
Набор тестов зависит от: Нет
Должен быть установлен до: File, GRUB, Kmod, Libelf, Man-DB и Udev
Необязательные зависимости: Нет

Zlib

Установка зависит от: Bash, Binutils, Coreutils, GCC, Glibc, Make и Sed
Требуется во время выполнения: Glibc
Набор тестов зависит от: Нет
Должен быть установлен до: File, Kmod, Libelf, Perl и Util-linux
Необязательные зависимости: Нет

Zstd

Установка зависит от: Binutils, Coreutils, GCC, Glibc, Gzip, Lz4, Make, Xz и Zlib
Требуется во время выполнения: Glibc
Набор тестов зависит от: Нет
Должен быть установлен до: Binutils, GCC, Libelf и Udev
Необязательные зависимости: Нет

Приложение D. Скрипты загрузки и настройки системы-20240825

Скрипты в приложении перечислены с указанием каталога, в котором они обычно находятся. Порядок такой: `/etc/rc.d/init.d` , `/etc/sysconfig` , `/etc/sysconfig/network-devices` , и `/etc/sysconfig/network-devices/services` . В каждом разделе файлы перечислены в том порядке, в котором они обычно вызываются.

D.1. /etc/rc.d/init.d/rc

Скрипт `rc` это первый скрипт, вызываемый системой инициализации init и инициирующий процесс загрузки.

```
#!/bin/bash
#############################################################################
# Begin rc
#
# Description : Main Run Level Control Script
#
# Authors     : Gerard Beekmans  - gerard@linuxfromscratch.org
#             : DJ Lucas - dj@linuxfromscratch.org
# Updates     : Bruce Dubbs - bdubbs@linuxfromscratch.org
#             : Pierre Labastie - pierre@linuxfromscratch.org
#
# Version     : LFS 7.0
#
# Notes       : Updates March 24th, 2022: new semantics of S/K files
#               - Instead of testing that S scripts were K scripts in the
#                 previous runlevel, test that they were not S scripts
#               - Instead of testing that K scripts were S scripts in the
#                 previous runlevel, test that they were not K scripts
#               - S scripts in runlevel 0 or 6 are now run with
#                "script start" (was "script stop" previously).
#############################################################################

. /lib/lsb/init-functions

print_error_msg()
{
   log_failure_msg
   # $i is set when called
   MSG="FAILURE:\n\nYou should not be reading this error message.\n\n"
   MSG="${MSG}It means that an unforeseen error took place in\n"
   MSG="${MSG}${i},\n"
   MSG="${MSG}which exited with a return value of ${error_value}.\n"

   MSG="${MSG}If you're able to track this error down to a bug in one of\n"
   MSG="${MSG}the files provided by the ${DISTRO_MINI} book,\n"
   MSG="${MSG}please be so kind to inform us at ${DISTRO_CONTACT}.\n"
   log_failure_msg "${MSG}"

   log_info_msg "Press Enter to continue..."
   wait_for_user
}

check_script_status()
{
   # $i is set when called
   if [ ! -f ${i} ]; then
      log_warning_msg "${i} is not a valid symlink."
      SCRIPT_STAT="1"
   fi

   if [ ! -x ${i} ]; then
```

```
      log_warning_msg "${i} is not executable, skipping."
      SCRIPT_STAT="1"
   fi
}

run()
{
   if [ -z $interactive ]; then
      ${1} ${2}
      return $?
   fi

   while true; do
      read -p "Run ${1} ${2} (Yes/no/continue)? " -n 1 runit
      echo

      case ${runit} in
         c | C)
            interactive=""
            ${i} ${2}
            ret=${?}
            break;
            ;;

         n | N)
            return 0
            ;;

         y | Y)
            ${i} ${2}
            ret=${?}
            break
            ;;
      esac
   done

   return $ret
}

# Read any local settings/overrides
[ -r /etc/sysconfig/rc.site ] && source /etc/sysconfig/rc.site

DISTRO=${DISTRO:-"Linux From Scratch"}
DISTRO_CONTACT=${DISTRO_CONTACT:-"lfs-dev@lists.linuxfromscratch.org (Registration required)"}
DISTRO_MINI=${DISTRO_MINI:-"LFS"}
IPROMPT=${IPROMPT:-"no"}

# These 3 signals will not cause our script to exit
trap "" INT QUIT TSTP

[ "${1}" != "" ] && runlevel=${1}

if [ "${runlevel}" == "" ]; then
   echo "Usage: ${0} <runlevel>" >&2
   exit 1
fi

previous=${PREVLEVEL}
[ "${previous}" == "" ] && previous=N

if [ ! -d /etc/rc.d/rc${runlevel}.d ]; then
   log_info_msg "/etc/rc.d/rc${runlevel}.d does not exist.\n"
   exit 1
fi

if [ "$runlevel" == "6" -o "$runlevel" == "0" ]; then IPROMPT="no"; fi
```

```
# Note: In ${LOGLEVEL:-7}, it is ':' 'dash' '7', not minus 7
if [ "$runlevel" == "S" ]; then
   [ -r /etc/sysconfig/console ] && source /etc/sysconfig/console
   dmesg -n "${LOGLEVEL:-7}"
fi

if [ "${IPROMPT}" == "yes" -a "${runlevel}" == "S" ]; then
   # The total length of the distro welcome string, without escape codes
   wlen=${wlen:-$(echo "Welcome to ${DISTRO}" | wc -c )}
   welcome_message=${welcome_message:-"Welcome to ${INFO}${DISTRO}${NORMAL}"}

   # The total length of the interactive string, without escape codes
   ilen=${ilen:-$(echo "Press 'I' to enter interactive startup" | wc -c )}
   i_message=${i_message:-"Press '${FAILURE}I${NORMAL}' to enter interactive startup"}


   # dcol and icol are spaces before the message to center the message
   # on screen. itime is the amount of wait time for the user to press a key
   wcol=$(( ( ${COLUMNS} - ${wlen} ) / 2 ))
   icol=$(( ( ${COLUMNS} - ${ilen} ) / 2 ))
   itime=${itime:-"3"}

   echo -e "\n\n"
   echo -e "\\033[${wcol}G${welcome_message}"
   echo -e "\\033[${icol}G${i_message}${NORMAL}"
   echo ""
   read -t "${itime}" -n 1 interactive 2>&1 > /dev/null
fi

# Make lower case
[ "${interactive}" == "I" ] && interactive="i"
[ "${interactive}" != "i" ] && interactive=""

# Read the state file if it exists from runlevel S
[ -r /run/interactive ] && source /run/interactive

# Stop all services marked as K, except if marked as K in the previous
# runlevel: it is the responsibility of the script to not try to kill
# a non running service
if [ "${previous}" != "N" ]; then
   for i in $(ls -v /etc/rc.d/rc${runlevel}.d/K* 2> /dev/null)
   do
      check_script_status
      if [ "${SCRIPT_STAT}" == "1" ]; then
         SCRIPT_STAT="0"
         continue
      fi

      suffix=${i#/etc/rc.d/rc${runlevel}.d/K[0-9][0-9]}
      [ -e /etc/rc.d/rc${previous}.d/K[0-9][0-9]$suffix ] && continue

      run ${i} stop
      error_value=${?}

      if [ "${error_value}" != "0" ]; then print_error_msg; fi
   done
fi

if [ "${previous}" == "N" ]; then export IN_BOOT=1; fi

if [ "$runlevel" == "6" -a -n "${FASTBOOT}" ]; then
   touch /fastboot
fi


# Start all services marked as S in this runlevel, except if marked as
# S in the previous runlevel
```

```
# it is the responsibility of the script to not try to start an already running
# service
for i in $( ls -v /etc/rc.d/rc${runlevel}.d/S* 2> /dev/null)
do

   if [ "${previous}" != "N" ]; then
      suffix=${i#/etc/rc.d/rc${runlevel}.d/S[0-9][0-9]}
      [ -e /etc/rc.d/rc${previous}.d/S[0-9][0-9]$suffix ] && continue
   fi

   check_script_status
   if [ "${SCRIPT_STAT}" == "1" ]; then
      SCRIPT_STAT="0"
      continue
   fi

   run ${i} start

   error_value=${?}

   if [ "${error_value}" != "0" ]; then print_error_msg; fi
done

# Store interactive variable on switch from runlevel S and remove if not
if [ "${runlevel}" == "S" -a "${interactive}" == "i" ]; then
    echo "interactive=\"i\"" > /run/interactive
else
    rm -f /run/interactive 2> /dev/null
fi

# Copy the boot log on initial boot only
if [ "${previous}" == "N" -a  "${runlevel}" != "S" ]; then
   cat $BOOTLOG >> /var/log/boot.log

   # Mark the end of boot
   echo "--------" >> /var/log/boot.log

   # Remove the temporary file
   rm -f $BOOTLOG 2> /dev/null
fi

# End rc
```

D.2. /lib/lsb/init-functions

```
#!/bin/sh
########################################################################
#
# Begin /lib/lsb/init-funtions
#
# Description : Run Level Control Functions
#
# Authors     : Gerard Beekmans - gerard@linuxfromscratch.org
#             : DJ Lucas - dj@linuxfromscratch.org
# Update      : Bruce Dubbs - bdubbs@linuxfromscratch.org
#
# Version     : LFS 7.0
#
# Notes       : With code based on Matthias Benkmann's simpleinit-msb
#               http://winterdrache.de/linux/newboot/index.html
#
#               The file should be located in /lib/lsb
#
########################################################################

## Environmental setup
# Setup default values for environment
```

```
umask 022
export PATH="/bin:/usr/bin:/sbin:/usr/sbin"

## Set color commands, used via echo
# Please consult `man console_codes for more information
# under the "ECMA-48 Set Graphics Rendition" section
#
# Warning: when switching from a 8bit to a 9bit font,
# the linux console will reinterpret the bold (1;) to
# the top 256 glyphs of the 9bit font.  This does
# not affect framebuffer consoles

NORMAL="\\033[0;39m"         # Standard console grey
SUCCESS="\\033[1;32m"        # Success is green
WARNING="\\033[1;33m"        # Warnings are yellow
FAILURE="\\033[1;31m"        # Failures are red
INFO="\\033[1;36m"           # Information is light cyan
BRACKET="\\033[1;34m"        # Brackets are blue

# Use a colored prefix
BMPREFIX="      "
SUCCESS_PREFIX="${SUCCESS}  *  ${NORMAL} "
FAILURE_PREFIX="${FAILURE}*****${NORMAL} "
WARNING_PREFIX="${WARNING} *** ${NORMAL} "
SKIP_PREFIX="${INFO}  S   ${NORMAL}"

SUCCESS_SUFFIX="${BRACKET}[${SUCCESS}  OK  ${BRACKET}]${NORMAL}"
FAILURE_SUFFIX="${BRACKET}[${FAILURE} FAIL ${BRACKET}]${NORMAL}"
WARNING_SUFFIX="${BRACKET}[${WARNING} WARN ${BRACKET}]${NORMAL}"
SKIP_SUFFIX="${BRACKET}[${INFO} SKIP ${BRACKET}]${NORMAL}"

BOOTLOG=/run/bootlog
KILLDELAY=3
SCRIPT_STAT="0"

# Set any user specified environment variables e.g. HEADLESS
[ -r /etc/sysconfig/rc.site ]  && . /etc/sysconfig/rc.site

# If HEADLESS is set, use that.
# If file descriptor 1 or 2 (stdout and stderr) is not open or
# does not refer to a terminal, consider the script headless.
[ ! -t 1 -o ! -t 2 ] && HEADLESS=${HEADLESS:-yes}

if [ "x$HEADLESS" != "xyes" ]
then
  ## Screen Dimensions
  # Find current screen size
  if [ -z "${COLUMNS}" ]; then
    COLUMNS=$(stty size)
    COLUMNS=${COLUMNS##* }
  fi
else
   COLUMNS=80
fi

# When using remote connections, such as a serial port, stty size returns 0
if [ "${COLUMNS}" = "0" ]; then
   COLUMNS=80
fi

## Measurements for positioning result messages
COL=$((${COLUMNS} - 8))
WCOL=$((${COL} - 2))

## Set Cursor Position Commands, used via echo
SET_COL="\\033[${COL}G"      # at the $COL char
SET_WCOL="\\033[${WCOL}G"    # at the $WCOL char
```

```
CURS_UP="\\033[1A\\033[0G"   # Up one line, at the 0'th char
CURS_ZERO="\\033[0G"

###############################################################################
# start_daemon()                                                              #
# Usage: start_daemon [-f] [-n nicelevel] [-p pidfile] pathname [args...]     #
#                                                                             #
# Purpose: This runs the specified program as a daemon                        #
#                                                                             #
# Inputs: -f: (force) run the program even if it is already running.          #
#         -n nicelevel: specify a nice level. See 'man nice(1)'.              #
#         -p pidfile: use the specified file to determine PIDs.               #
#         pathname: the complete path to the specified program                #
#         args: additional arguments passed to the program (pathname)         #
#                                                                             #
# Return values (as defined by LSB exit codes):                               #
#       0 - program is running or service is OK                               #
#       1 - generic or unspecified error                                      #
#       2 - invalid or excessive argument(s)                                  #
#       5 - program is not installed                                          #
###############################################################################
start_daemon()
{
    local force=""
    local nice="0"
    local pidfile=""
    local pidlist=""
    local retval=""

    # Process arguments
    while true
    do
        case "${1}" in

            -f)
                force="1"
                shift 1
                ;;

            -n)
                nice="${2}"
                shift 2
                ;;

            -p)
                pidfile="${2}"
                shift 2
                ;;

            -*)
                return 2
                ;;

            *)
                program="${1}"
                break
                ;;
        esac
    done

    # Check for a valid program
    if [ ! -e "${program}" ]; then return 5; fi

    # Execute
    if [ -z "${force}" ]; then
        if [ -z "${pidfile}" ]; then
            # Determine the pid by discovery
```

```
            pidlist=`pidofproc "${1}"`
            retval="${?}"
        else
            # The PID file contains the needed PIDs
            # Note that by LSB requirement, the path must be given to pidofproc,
            # however, it is not used by the current implementation or standard.
            pidlist=`pidofproc -p "${pidfile}" "${1}"`
            retval="${?}"
        fi

        # Return a value ONLY
        # It is the init script's (or distribution's functions) responsibility
        # to log messages!
        case "${retval}" in

            0)
                # Program is already running correctly, this is a
                # successful start.
                return 0
                ;;

            1)
                # Program is not running, but an invalid pid file exists
                # remove the pid file and continue
                rm -f "${pidfile}"
                ;;

            3)
                # Program is not running and no pidfile exists
                # do nothing here, let start_deamon continue.
                ;;

            *)
                # Others as returned by status values shall not be interpreted
                # and returned as an unspecified error.
                return 1
                ;;
        esac
    fi

    # Do the start!
    nice -n "${nice}" "${@}"
}

########################################################################
# killproc()                                                           #
# Usage: killproc [-p pidfile] pathname [signal]                       #
#                                                                      #
# Purpose: Send control signals to running processes                   #
#                                                                      #
# Inputs: -p pidfile, uses the specified pidfile                       #
#         pathname, pathname to the specified program                  #
#         signal, send this signal to pathname                         #
#                                                                      #
# Return values (as defined by LSB exit codes):                       #
#       0 - program (pathname) has stopped/is already stopped or a     #
#           running program has been sent specified signal and stopped #
#           successfully                                               #
#       1 - generic or unspecified error                               #
#       2 - invalid or excessive argument(s)                           #
#       5 - program is not installed                                   #
#       7 - program is not running and a signal was supplied           #
########################################################################
killproc()
{
    local pidfile
    local program
```

```
    local prefix
    local progname
    local signal="-TERM"
    local fallback="-KILL"
    local nosig
    local pidlist
    local retval
    local pid
    local delay="30"
    local piddead
    local dtime

    # Process arguments
    while true; do
        case "${1}" in
            -p)
                pidfile="${2}"
                shift 2
                ;;

             *)
                 program="${1}"
                 if [ -n "${2}" ]; then
                     signal="${2}"
                     fallback=""
                 else
                     nosig=1
                 fi

                 # Error on additional arguments
                 if [ -n "${3}" ]; then
                     return 2
                 else
                     break
                 fi
                 ;;
        esac
    done

    # Check for a valid program
    if [ ! -e "${program}" ]; then return 5; fi

    # Check for a valid signal
    check_signal "${signal}"
    if [ "${?}" -ne "0" ]; then return 2; fi

    # Get a list of pids
    if [ -z "${pidfile}" ]; then
        # determine the pid by discovery
        pidlist=`pidofproc "${1}"`
        retval="${?}"
    else
        # The PID file contains the needed PIDs
        # Note that by LSB requirement, the path must be given to pidofproc,
        # however, it is not used by the current implementation or standard.
        pidlist=`pidofproc -p "${pidfile}" "${1}"`
        retval="${?}"
    fi

    # Return a value ONLY
    # It is the init script's (or distribution's functions) responsibility
    # to log messages!
    case "${retval}" in

        0)
            # Program is running correctly
            # Do nothing here, let killproc continue.
```

```
            ;;

        1)
            # Program is not running, but an invalid pid file exists
            # Remove the pid file.

            progname=${program##*/}

            if [[ -e "/run/${progname}.pid" ]]; then
                pidfile="/run/${progname}.pid"
                rm -f "${pidfile}"
            fi

            # This is only a success if no signal was passed.
            if [ -n "${nosig}" ]; then
                return 0
            else
                return 7
            fi
            ;;

        3)
            # Program is not running and no pidfile exists
            # This is only a success if no signal was passed.
            if [ -n "${nosig}" ]; then
                return 0
            else
                return 7
            fi
            ;;

        *)
            # Others as returned by status values shall not be interpreted
            # and returned as an unspecified error.
            return 1
            ;;
    esac

    # Perform different actions for exit signals and control signals
    check_sig_type "${signal}"

    if [ "${?}" -eq "0" ]; then # Signal is used to terminate the program

        # Account for empty pidlist (pid file still exists and no
        # signal was given)
        if [ "${pidlist}" != "" ]; then

            # Kill the list of pids
            for pid in ${pidlist}; do

                kill -0 "${pid}" 2> /dev/null

                if [ "${?}" -ne "0" ]; then
                    # Process is dead, continue to next and assume all is well
                    continue
                else
                    kill "${signal}" "${pid}" 2> /dev/null

                    # Wait up to ${delay}/10 seconds to for "${pid}" to
                    # terminate in 10ths of a second

                    while [ "${delay}" -ne "0" ]; do
                        kill -0 "${pid}" 2> /dev/null || piddead="1"
                        if [ "${piddead}" = "1" ]; then break; fi
                        sleep 0.1
                        delay="$(( ${delay} - 1 ))"
                    done
```

```
                    # If a fallback is set, and program is still running, then
                    # use the fallback
                    if [ -n "${fallback}" -a "${piddead}" != "1" ]; then
                        kill "${fallback}" "${pid}" 2> /dev/null
                        sleep 1
                        # Check again, and fail if still running
                        kill -0 "${pid}" 2> /dev/null && return 1
                    fi
                fi
            done
        fi

        # Check for and remove stale PID files.
        if [ -z "${pidfile}" ]; then
            # Find the basename of $program
            prefix=`echo "${program}" | sed 's/[^/]*$//'`
            progname=`echo "${program}" | sed "s@${prefix}@@"`

            if [ -e "/run/${progname}.pid" ]; then
                rm -f "/run/${progname}.pid" 2> /dev/null
            fi
        else
            if [ -e "${pidfile}" ]; then rm -f "${pidfile}" 2> /dev/null; fi
        fi

    # For signals that do not expect a program to exit, simply
    # let kill do its job, and evaluate kill's return for value

    else # check_sig_type - signal is not used to terminate program
        for pid in ${pidlist}; do
            kill "${signal}" "${pid}"
            if [ "${?}" -ne "0" ]; then return 1; fi
        done
    fi
}

##############################################################################
# pidofproc()                                                                #
# Usage: pidofproc [-p pidfile] pathname                                     #
#                                                                            #
# Purpose: This function returns one or more pid(s) for a particular daemon  #
#                                                                            #
# Inputs: -p pidfile, use the specified pidfile instead of pidof             #
#         pathname, path to the specified program                            #
#                                                                            #
# Return values (as defined by LSB status codes):                            #
#       0 - Success (PIDs to stdout)                                         #
#       1 - Program is dead, PID file still exists (remaining PIDs output)   #
#       3 - Program is not running (no output)                               #
##############################################################################
pidofproc()
{
    local pidfile
    local program
    local prefix
    local progname
    local pidlist
    local lpids
    local exitstatus="0"

    # Process arguments
    while true; do
        case "${1}" in

            -p)
                pidfile="${2}"
```

```
                shift 2
                ;;

            *)
                program="${1}"
                if [ -n "${2}" ]; then
                    # Too many arguments
                    # Since this is status, return unknown
                    return 4
                else
                    break
                fi
                ;;
        esac
    done

    # If a PID file is not specified, try and find one.
    if [ -z "${pidfile}" ]; then
        # Get the program's basename
        prefix=`echo "${program}" | sed 's/[^/]*$//'`

        if [ -z "${prefix}" ]; then
           progname="${program}"
        else
           progname=`echo "${program}" | sed "s@${prefix}@@"`
        fi

        # If a PID file exists with that name, assume that is it.
        if [ -e "/run/${progname}.pid" ]; then
            pidfile="/run/${progname}.pid"
        fi
    fi

    # If a PID file is set and exists, use it.
    if [ -n "${pidfile}" -a -e "${pidfile}" ]; then
        # Use the value in the first line of the pidfile
        pidlist=`/bin/head -n1 "${pidfile}"`
    else
        # Use pidof
        pidlist=`pidof "${program}"`
    fi

    # Figure out if all listed PIDs are running.
    for pid in ${pidlist}; do
        kill -0 ${pid} 2> /dev/null

        if [ "${?}" -eq "0" ]; then
            lpids="${lpids}${pid} "
        else
            exitstatus="1"
        fi
    done

    if [ -z "${lpids}" -a ! -f "${pidfile}" ]; then
        return 3
    else
        echo "${lpids}"
        return "${exitstatus}"
    fi
}

################################################################################
# statusproc()                                                                 #
# Usage: statusproc [-p pidfile] pathname                                      #
#                                                                              #
# Purpose: This function prints the status of a particular daemon to stdout    #
#                                                                              #
```

```
# Inputs: -p pidfile, use the specified pidfile instead of pidof               #
#         pathname, path to the specified program                              #
#                                                                              #
# Return values:                                                               #
#       0 - Status printed                                                     #
#       1 - Input error. The daemon to check was not specified.                #
################################################################################
statusproc()
{
   local pidfile
   local pidlist

   if [ "${#}" = "0" ]; then
      echo "Usage: statusproc [-p pidfle] {program}"
      exit 1
   fi

   # Process arguments
   while true; do
       case "${1}" in

           -p)
               pidfile="${2}"
               shift 2
               ;;

           *)
               if [ -n "${2}" ]; then
                   echo "Too many arguments"
                   return 1
               else
                   break
               fi
               ;;
       esac
   done

   if [ -n "${pidfile}" ]; then
      pidlist=`pidofproc -p "${pidfile}" $@`
   else
      pidlist=`pidofproc $@`
   fi

   # Trim trailing blanks
   pidlist=`echo "${pidlist}" | sed -r 's/ +$//'`

   base="${1##*/}"

   if [ -n "${pidlist}" ]; then
      /bin/echo -e "${INFO}${base} is running with Process" \
         "ID(s) ${pidlist}.${NORMAL}"
   else
      if [ -n "${base}" -a -e "/run/${base}.pid" ]; then
         /bin/echo -e "${WARNING}${1} is not running but" \
            "/run/${base}.pid exists.${NORMAL}"
      else
         if [ -n "${pidfile}" -a -e "${pidfile}" ]; then
            /bin/echo -e "${WARNING}${1} is not running" \
               "but ${pidfile} exists.${NORMAL}"
         else
            /bin/echo -e "${INFO}${1} is not running.${NORMAL}"
         fi
      fi
   fi
}

################################################################################
```

```
# timespec()                                                                  #
#                                                                             #
# Purpose: An internal utility function to format a timestamp                 #
#          a boot log file.  Sets the STAMP variable.                         #
#                                                                             #
# Return value: Not used                                                      #
###############################################################################
timespec()
{
   STAMP="$(echo `date +"%b %d %T %:z"` `hostname`) "
   return 0
}

###############################################################################
# log_success_msg()                                                           #
# Usage: log_success_msg ["message"]                                          #
#                                                                             #
# Purpose: Print a successful status message to the screen and                #
#          a boot log file.                                                   #
#                                                                             #
# Inputs: $@ - Message                                                        #
#                                                                             #
# Return values: Not used                                                     #
###############################################################################
log_success_msg()
{
    if [ "x$HEADLESS" != "xyes" ]
    then
      /bin/echo -n -e "${BMPREFIX}${@}"
      /bin/echo -e "${CURS_ZERO}${SUCCESS_PREFIX}${SET_COL}${SUCCESS_SUFFIX}"
    else
      logmessage=`echo "${@}" | sed 's/\\\033[^a-zA-Z]*.//g'`
      /bin/echo -e "${logmessage} OK"
    fi
    # Strip non-printable characters from log file
    logmessage=`echo "${@}" | sed 's/\\\033[^a-zA-Z]*.//g'`

    timespec
    /bin/echo -e "${STAMP} ${logmessage} OK" >> ${BOOTLOG}

    return 0
}

log_success_msg2()
{
    if [ "x$HEADLESS" != "xyes" ]
    then
      /bin/echo -n -e "${BMPREFIX}${@}"
      /bin/echo -e "${CURS_ZERO}${SUCCESS_PREFIX}${SET_COL}${SUCCESS_SUFFIX}"
    else
      echo " OK"
    fi

    echo " OK" >> ${BOOTLOG}

    return 0
}

###############################################################################
# log_failure_msg()                                                           #
# Usage: log_failure_msg ["message"]                                          #
#                                                                             #
# Purpose: Print a failure status message to the screen and                   #
#          a boot log file.                                                   #
#                                                                             #
# Inputs: $@ - Message                                                        #
#                                                                             #
```

```
# Return values: Not used                                                      #
###############################################################################
log_failure_msg()
{
    if [ "x$HEADLESS" != "xyes" ]
    then
      /bin/echo -n -e "${BMPREFIX}${@}"
      /bin/echo -e "${CURS_ZERO}${FAILURE_PREFIX}${SET_COL}${FAILURE_SUFFIX}"
    else
      logmessage=`echo "${@}" | sed 's/\\\033[^a-zA-Z]*.//g'`
      /bin/echo -e "${logmessage} FAIL"
    fi

    # Strip non-printable characters from log file

    timespec
    logmessage=`echo "${@}" | sed 's/\\\033[^a-zA-Z]*.//g'`
    /bin/echo -e "${STAMP} ${logmessage} FAIL" >> ${BOOTLOG}

    return 0
}

log_failure_msg2()
{
    if [ "x$HEADLESS" != "xyes" ]
    then
      /bin/echo -n -e "${BMPREFIX}${@}"
      /bin/echo -e "${CURS_ZERO}${FAILURE_PREFIX}${SET_COL}${FAILURE_SUFFIX}"
    else
      echo "FAIL"
    fi

    echo "FAIL" >> ${BOOTLOG}

    return 0
}

###############################################################################
# log_warning_msg()                                                           #
# Usage: log_warning_msg ["message"]                                          #
#                                                                             #
# Purpose: Print a warning status message to the screen and                   #
#          a boot log file.                                                   #
#                                                                             #
# Return values: Not used                                                     #
###############################################################################
log_warning_msg()
{
    if [ "x$HEADLESS" != "xyes" ]
    then
      /bin/echo -n -e "${BMPREFIX}${@}"
      /bin/echo -e "${CURS_ZERO}${WARNING_PREFIX}${SET_COL}${WARNING_SUFFIX}"
    else
      logmessage=`echo "${@}" | sed 's/\\\033[^a-zA-Z]*.//g'`
      /bin/echo -e "${logmessage} WARN"
    fi

    # Strip non-printable characters from log file
    logmessage=`echo "${@}" | sed 's/\\\033[^a-zA-Z]*.//g'`
    timespec
    /bin/echo -e "${STAMP} ${logmessage} WARN" >> ${BOOTLOG}

    return 0
}

log_skip_msg()
{
```

```
    if [ "x$HEADLESS" != "xyes" ]
    then
      /bin/echo -n -e "${BMPREFIX}${@}"
      /bin/echo -e "${CURS_ZERO}${SKIP_PREFIX}${SET_COL}${SKIP_SUFFIX}"
    else
      logmessage=`echo "${@}" | sed 's/\\\033[^a-zA-Z]*.//g'`
      /bin/echo "SKIP"
    fi

    # Strip non-printable characters from log file
    logmessage=`echo "${@}" | sed 's/\\\033[^a-zA-Z]*.//g'`
    /bin/echo "SKIP" >> ${BOOTLOG}

    return 0
}

##############################################################################
# log_info_msg()                                                             #
# Usage: log_info_msg message                                                #
#                                                                            #
# Purpose: Print an information message to the screen and                    #
#          a boot log file.  Does not print a trailing newline character.    #
#                                                                            #
# Return values: Not used                                                    #
##############################################################################
log_info_msg()
{
    if [ "x$HEADLESS" != "xyes" ]
    then
      /bin/echo -n -e "${BMPREFIX}${@}"
    else
      logmessage=`echo "${@}" | sed 's/\\\033[^a-zA-Z]*.//g'`
      /bin/echo -n -e "${logmessage}"
    fi

    # Strip non-printable characters from log file
    logmessage=`echo "${@}" | sed 's/\\\033[^a-zA-Z]*.//g'`
    timespec
    /bin/echo -n -e "${STAMP} ${logmessage}" >> ${BOOTLOG}

    return 0
}

log_info_msg2()
{
    if [ "x$HEADLESS" != "xyes" ]
    then
      /bin/echo -n -e "${@}"
    else
      logmessage=`echo "${@}" | sed 's/\\\033[^a-zA-Z]*.//g'`
      /bin/echo -n -e "${logmessage}"
    fi

    # Strip non-printable characters from log file
    logmessage=`echo "${@}" | sed 's/\\\033[^a-zA-Z]*.//g'`
    /bin/echo -n -e "${logmessage}" >> ${BOOTLOG}

    return 0
}

##############################################################################
# evaluate_retval()                                                          #
# Usage: Evaluate a return value and print success or failure as appropriate #
#                                                                            #
# Purpose: Convenience function to terminate an info message                 #
#                                                                            #
# Return values: Not used                                                    #
```

```
##############################################################################
evaluate_retval()
{
   local error_value="${?}"

   if [ ${error_value} = 0 ]; then
      log_success_msg2
   else
      log_failure_msg2
   fi
}

##############################################################################
# check_signal()                                                             #
# Usage: check_signal [ -{signal} ]                                          #
#                                                                            #
# Purpose: Check for a valid signal.  This is not defined by any LSB draft,  #
#          however, it is required to check the signals to determine if the  #
#          signals chosen are invalid arguments to the other functions.      #
#                                                                            #
# Inputs: Accepts a single string value in the form of -{signal}             #
#                                                                            #
# Return values:                                                             #
#       0 - Success (signal is valid                                         #
#       1 - Signal is not valid                                              #
##############################################################################
check_signal()
{
    local valsig

    # Add error handling for invalid signals
    valsig=" -ALRM -HUP -INT -KILL -PIPE -POLL -PROF -TERM -USR1 -USR2"
    valsig="${valsig} -VTALRM -STKFLT -PWR -WINCH -CHLD -URG -TSTP -TTIN"
    valsig="${valsig} -TTOU -STOP -CONT -ABRT -FPE -ILL -QUIT -SEGV -TRAP"
    valsig="${valsig} -SYS -EMT -BUS -XCPU -XFSZ -0 -1 -2 -3 -4 -5 -6 -8 -9"
    valsig="${valsig} -11 -13 -14 -15 "

    echo "${valsig}" | grep -- " ${1} " > /dev/null

    if [ "${?}" -eq "0" ]; then
        return 0
    else
        return 1
    fi
}

##############################################################################
# check_sig_type()                                                           #
# Usage: check_signal [ -{signal} | {signal} ]                               #
#                                                                            #
# Purpose: Check if signal is a program termination signal or a control signal #
#          This is not defined by any LSB draft, however, it is required to  #
#          check the signals to determine if they are intended to end a      #
#          program or simply to control it.                                  #
#                                                                            #
# Inputs: Accepts a single string value in the form or -{signal} or {signal} #
#                                                                            #
# Return values:                                                             #
#       0 - Signal is used for program termination                           #
#       1 - Signal is used for program control                               #
##############################################################################
check_sig_type()
{
    local valsig

    # The list of termination signals (limited to generally used items)
    valsig=" -ALRM -INT -KILL -TERM -PWR -STOP -ABRT -QUIT -2 -3 -6 -9 -14 -15 "
```

```
    echo "${valsig}" | grep -- " ${1} " > /dev/null

    if [ "${?}" -eq "0" ]; then
        return 0
    else
        return 1
    fi
}

################################################################################
# wait_for_user()                                                              #
#                                                                              #
# Purpose: Wait for the user to respond if not a headless system               #
#                                                                              #
################################################################################
wait_for_user()
{
   # Wait for the user by default
   [ "${HEADLESS=0}" = "0" ] && read ENTER
   return 0
}

################################################################################
# is_true()                                                                    #
#                                                                              #
# Purpose: Utility to test if a variable is true | yes | 1                     #
#                                                                              #
################################################################################
is_true()
{
   [ "$1" = "1" ] || [ "$1" = "yes" ] || [ "$1" = "true" ] ||  [ "$1" = "y" ] ||
   [ "$1" = "t" ]
}

# End /lib/lsb/init-functions
```

D.3. /etc/rc.d/init.d/mountvirtfs

```
#!/bin/sh
########################################################################
# Begin mountvirtfs
#
# Description : Ensure proc, sysfs, run, and dev are mounted
#
# Authors     : Gerard Beekmans - gerard@linuxfromscratch.org
#               DJ Lucas - dj@linuxfromscratch.org
# Update      : Bruce Dubbs - bdubbs@linuxfromscratch.org
#               Xi Ruoyao - xry111@xry111.site
#
# Version     : LFS 12.0
#
########################################################################

### BEGIN INIT INFO
# Provides:            mountvirtfs
# Required-Start:      $first
# Should-Start:
# Required-Stop:
# Should-Stop:
# Default-Start:       S
# Default-Stop:
# Short-Description:   Mounts various special fs needed at start
# Description:         Mounts /sys and /proc virtual (kernel) filesystems.
#                      Mounts /run (tmpfs) and /dev (devtmpfs).
#                      This is done only if they are not already mounted.
#                      with the kernel config proposed in the book, dev
```

```
#                       should be automatically mounted by the kernel.
# X-LFS-Provided-By:    LFS
### END INIT INFO

. /lib/lsb/init-functions

case "${1}" in
   start)
      # Make sure /run is available before logging any messages
      if ! mountpoint /run >/dev/null; then
         mount /run || failed=1
      fi

      mkdir -p   /run/lock
      chmod 1777 /run/lock

      log_info_msg "Mounting virtual file systems: ${INFO}/run"

      if ! mountpoint /proc >/dev/null; then
         log_info_msg2 " ${INFO}/proc"
         mount -o nosuid,noexec,nodev /proc || failed=1
      fi

      if ! mountpoint /sys >/dev/null; then
         log_info_msg2 " ${INFO}/sys"
         mount -o nosuid,noexec,nodev /sys || failed=1
      fi

      if ! mountpoint /dev >/dev/null; then
         log_info_msg2 " ${INFO}/dev"
         mount -o mode=0755,nosuid /dev  || failed=1
      fi

      mkdir -p /dev/shm
      log_info_msg2 " ${INFO}/dev/shm"
      mount -o nosuid,nodev /dev/shm || failed=1

      mkdir -p /sys/fs/cgroup
      log_info_msg2 " ${INFO}/sys/fs/cgroup"
      mount -o nosuid,noexec,nodev /sys/fs/cgroup || failed=1

      (exit ${failed})
      evaluate_retval
      if [ "${failed}" = 1 ]; then
         exit 1
      fi

      log_info_msg "Create symlinks in /dev targeting /proc: ${INFO}/dev/stdin"
      ln -sf /proc/self/fd/0 /dev/stdin  || failed=1

      log_info_msg2 " ${INFO}/dev/stdout"
      ln -sf /proc/self/fd/1 /dev/stdout || failed=1

      log_info_msg2 " ${INFO}/dev/stderr"
      ln -sf /proc/self/fd/2 /dev/stderr || failed=1

      log_info_msg2 " ${INFO}/dev/fd"
      ln -sfn /proc/self/fd  /dev/fd     || failed=1

      if [ -e /proc/kcore ]; then
         log_info_msg2 " ${INFO}/dev/core"
         ln -sf /proc/kcore  /dev/core   || failed=1
      fi

      (exit ${failed})
      evaluate_retval
      exit $failed
```

```
      ;;

   *)
      echo "Usage: ${0} {start}"
      exit 1
      ;;
esac

# End mountvirtfs
```

D.4. /etc/rc.d/init.d/modules

```
#!/bin/sh
########################################################################
# Begin modules
#
# Description : Module auto-loading script
#
# Authors     : Zack Winkles
#               DJ Lucas - dj@linuxfromscratch.org
# Update      : Bruce Dubbs - bdubbs@linuxfromscratch.org
#
# Version     : LFS 7.0
#
########################################################################

### BEGIN INIT INFO
# Provides:            modules
# Required-Start:      mountvirtfs
# Should-Start:
# Required-Stop:
# Should-Stop:
# Default-Start:       S
# Default-Stop:
# Short-Description:   Loads required modules.
# Description:         Loads modules listed in /etc/sysconfig/modules.
# X-LFS-Provided-By:   LFS
### END INIT INFO

# Assure that the kernel has module support.
[ -e /proc/modules ] || exit 0

. /lib/lsb/init-functions

case "${1}" in
   start)
      # Exit if there's no modules file or there are no
      # valid entries
      [ -r /etc/sysconfig/modules ]             || exit 0
      grep -E -qv '^($|#)' /etc/sysconfig/modules || exit 0

      log_info_msg "Loading modules:"

      # Only try to load modules if the user has actually given us
      # some modules to load.

      while read module args; do

         # Ignore comments and blank lines.
         case "$module" in
            ""|"#"*) continue ;;
         esac

         # Attempt to load the module, passing any arguments provided.
         modprobe ${module} ${args} >/dev/null

         # Print the module name if successful, otherwise take note.
```

```
         if [ $? -eq 0 ]; then
            log_info_msg2 " ${module}"
         else
            failedmod="${failedmod} ${module}"
         fi
      done < /etc/sysconfig/modules

      # Print a message about successfully loaded modules on the correct line.
      log_success_msg2

      # Print a failure message with a list of any modules that
      # may have failed to load.
      if [ -n "${failedmod}" ]; then
         log_failure_msg "Failed to load modules:${failedmod}"
         exit 1
      fi
      ;;

   *)
      echo "Usage: ${0} {start}"
      exit 1
      ;;
esac

exit 0

# End modules
```

D.5. /etc/rc.d/init.d/udev

```
#!/bin/sh
########################################################################
# Begin udev
#
# Description : Udev cold-plugging script
#
# Authors     : Zack Winkles, Alexander E. Patrakov
#               DJ Lucas - dj@linuxfromscratch.org
# Update      : Bruce Dubbs - bdubbs@linuxfromscratch.org
#               Xi Ruoyao - xry111@xry111.site
#
# Version     : LFS 12.0
#
########################################################################

### BEGIN INIT INFO
# Provides:            udev $time
# Required-Start:      localnet
# Should-Start:        modules
# Required-Stop:
# Should-Stop:
# Default-Start:       S
# Default-Stop:
# Short-Description:   Populates /dev with device nodes.
# Description:         Mounts a tempfs on /dev and starts the udevd daemon.
#                      Device nodes are created as defined by udev.
# X-LFS-Provided-By:   LFS
### END INIT INFO

. /lib/lsb/init-functions

case "${1}" in
   start)
      log_info_msg "Populating /dev with device nodes... "
      if ! grep -q '[[:space:]]sysfs' /proc/mounts; then
         log_failure_msg2
         msg="FAILURE:\n\nUnable to create "
```

```
         msg="${msg}devices without a SysFS filesystem\n\n"
         msg="${msg}After you press Enter, this system "
         msg="${msg}will be halted and powered off.\n\n"
         log_info_msg "$msg"
         log_info_msg "Press Enter to continue..."
         wait_for_user
         /etc/rc.d/init.d/halt start
      fi

      # Start the udev daemon to continually watch for, and act on,
      # uevents
      SYSTEMD_LOG_TARGET=kmsg /sbin/udevd --daemon

      # Now traverse /sys in order to "coldplug" devices that have
      # already been discovered
      /bin/udevadm trigger --action=add    --type=subsystems
      /bin/udevadm trigger --action=add    --type=devices
      /bin/udevadm trigger --action=change --type=devices

      # Now wait for udevd to process the uevents we triggered
      if ! is_true "$OMIT_UDEV_SETTLE"; then
         /bin/udevadm settle
      fi

      # If any LVM based partitions are on the system, ensure they
      # are activated so they can be used.
      if [ -x /sbin/vgchange ]; then /sbin/vgchange -a y >/dev/null; fi

      log_success_msg2
      ;;

   *)
      echo "Usage ${0} {start}"
      exit 1
      ;;
esac

exit 0

# End udev
```

D.6. /etc/rc.d/init.d/swap

```
#!/bin/sh
########################################################################
# Begin swap
#
# Description : Swap Control Script
#
# Authors     : Gerard Beekmans - gerard@linuxfromscratch.org
#               DJ Lucas - dj@linuxfromscratch.org
# Update      : Bruce Dubbs - bdubbs@linuxfromscratch.org
#
# Version     : LFS 7.0
#
########################################################################

### BEGIN INIT INFO
# Provides:            swap
# Required-Start:      udev
# Should-Start:        modules
# Required-Stop:       localnet
# Should-Stop:         $local_fs
# Default-Start:       S
# Default-Stop:        0 6
# Short-Description:   Activates and deactivates swap partitions.
# Description:         Activates and deactivates swap partitions defined in
```

```
#                       /etc/fstab.
# X-LFS-Provided-By:    LFS
### END INIT INFO

. /lib/lsb/init-functions

case "${1}" in
   start)
      log_info_msg "Activating all swap files/partitions..."
      swapon -a
      evaluate_retval
      ;;

   stop)
      log_info_msg "Deactivating all swap files/partitions..."
      swapoff -a
      evaluate_retval
      ;;

   restart)
      ${0} stop
      sleep 1
      ${0} start
      ;;

   status)
      log_success_msg "Retrieving swap status."
      swapon -s
      ;;

   *)
      echo "Usage: ${0} {start|stop|restart|status}"
      exit 1
      ;;
esac

exit 0

# End swap
```

D.7. /etc/rc.d/init.d/setclock

```
#!/bin/sh
########################################################################
# Begin setclock
#
# Description : Setting Linux Clock
#
# Authors     : Gerard Beekmans - gerard@linuxfromscratch.org
#               DJ Lucas - dj@linuxfromscratch.org
# Update      : Bruce Dubbs - bdubbs@linuxfromscratch.org
#
# Version     : LFS 7.0
#
########################################################################

### BEGIN INIT INFO
# Provides:
# Required-Start:
# Should-Start:         modules
# Required-Stop:
# Should-Stop:          $syslog
# Default-Start:        S
# Default-Stop:
# Short-Description:    Stores and restores time from the hardware clock
# Description:          On boot, system time is obtained from hwclock.  The
#                       hardware clock can also be set on shutdown.
```

```
# X-LFS-Provided-By:   LFS
### END INIT INFO

. /lib/lsb/init-functions

[ -r /etc/sysconfig/clock ] && . /etc/sysconfig/clock


case "${UTC}" in
   yes|true|1)
      CLOCKPARAMS="${CLOCKPARAMS} --utc"
      ;;

   no|false|0)
      CLOCKPARAMS="${CLOCKPARAMS} --localtime"
      ;;

esac

case ${1} in
   start)
      hwclock --hctosys ${CLOCKPARAMS} >/dev/null
      ;;

   stop)
      log_info_msg "Setting hardware clock..."
      hwclock --systohc ${CLOCKPARAMS} >/dev/null
      evaluate_retval
      ;;

   *)
      echo "Usage: ${0} {start|stop}"
      exit 1
      ;;

esac

exit 0
```

D.8. /etc/rc.d/init.d/checkfs

```
#!/bin/sh
########################################################################
# Begin checkfs
#
# Description : File System Check
#
# Authors     : Gerard Beekmans - gerard@linuxfromscratch.org
#               A. Luebke - luebke@users.sourceforge.net
#               DJ Lucas - dj@linuxfromscratch.org
# Update      : Bruce Dubbs - bdubbs@linuxfromscratch.org
#
# Version     : LFS 7.0
#
# Based on checkfs script from LFS-3.1 and earlier.
#
# From man fsck
# 0    - No errors
# 1    - File system errors corrected
# 2    - System should be rebooted
# 4    - File system errors left uncorrected
# 8    - Operational error
# 16   - Usage or syntax error
# 32   - Fsck canceled by user request
# 128  - Shared library error
#
########################################################################
```

```
### BEGIN INIT INFO
# Provides:            checkfs
# Required-Start:      udev swap
# Should-Start:
# Required-Stop:
# Should-Stop:
# Default-Start:       S
# Default-Stop:
# Short-Description:   Checks local filesystems before mounting.
# Description:         Checks local filesystems before mounting.
# X-LFS-Provided-By:   LFS
### END INIT INFO

. /lib/lsb/init-functions

case "${1}" in
   start)
      if [ -f /fastboot ]; then
         msg="/fastboot found, will omit "
         msg="${msg} file system checks as requested.\n"
         log_info_msg "${msg}"
         exit 0
      fi

      log_info_msg "Mounting root file system in read-only mode... "
      mount -n -o remount,ro / >/dev/null

      if [ ${?} != 0 ]; then
         log_failure_msg2
         msg="\n\nCannot check root "
         msg="${msg}filesystem because it could not be mounted "
         msg="${msg}in read-only mode.\n\n"
         msg="${msg}After you press Enter, this system will be "
         msg="${msg}halted and powered off.\n\n"
         log_failure_msg "${msg}"

         log_info_msg "Press Enter to continue..."
         wait_for_user
         /etc/rc.d/init.d/halt start
      else
         log_success_msg2
      fi

      if [ -f /forcefsck ]; then
         msg="/forcefsck found, forcing file"
         msg="${msg} system checks as requested."
         log_success_msg "$msg"
         options="-f"
      else
         options=""
      fi

      log_info_msg "Checking file systems..."
      # Note: -a option used to be -p; but this fails e.g. on fsck.minix
      if is_true "$VERBOSE_FSCK"; then
        fsck ${options} -a -A -C -T
      else
        fsck ${options} -a -A -C -T >/dev/null
      fi

      error_value=${?}

      if [ "${error_value}" = 0 ]; then
         log_success_msg2
      fi
```

```
      if [ "${error_value}" = 1 ]; then
         msg="\nWARNING:\n\nFile system errors "
         msg="${msg}were found and have been corrected.\n"
         msg="${msg}      You may want to double-check that "
         msg="${msg}everything was fixed properly."
         log_warning_msg "$msg"
      fi

      if [ "${error_value}" = 2 -o "${error_value}" = 3 ]; then
         msg="\nWARNING:\n\nFile system errors "
         msg="${msg}were found and have been "
         msg="${msg}corrected, but the nature of the "
         msg="${msg}errors require this system to be rebooted.\n\n"
         msg="${msg}After you press enter, "
         msg="${msg}this system will be rebooted\n\n"
         log_failure_msg "$msg"

         log_info_msg "Press Enter to continue..."
         wait_for_user
         reboot -f
      fi

      if [ "${error_value}" -gt 3 -a "${error_value}" -lt 16 ]; then
         msg="\nFAILURE:\n\nFile system errors "
         msg="${msg}were encountered that could not be "
         msg="${msg}fixed automatically.\nThis system "
         msg="${msg}cannot continue to boot and will "
         msg="${msg}therefore be halted until those "
         msg="${msg}errors are fixed manually by a "
         msg="${msg}System Administrator.\n\n"
         msg="${msg}After you press Enter, this system will be "
         msg="${msg}halted and powered off.\n\n"
         log_failure_msg "$msg"

         log_info_msg "Press Enter to continue..."
         wait_for_user
         /etc/rc.d/init.d/halt start
      fi

      if [ "${error_value}" -ge 16 ]; then
         msg="FAILURE:\n\nUnexpected failure "
         msg="${msg}running fsck.  Exited with error "
         msg="${msg} code: ${error_value}.\n"
         log_info_msg $msg
         exit ${error_value}
      fi

      exit 0
      ;;
   *)
      echo "Usage: ${0} {start}"
      exit 1
      ;;
esac

# End checkfs
```

D.9. /etc/rc.d/init.d/mountfs

```
#!/bin/sh
########################################################################
# Begin mountfs
#
# Description : File System Mount Script
#
# Authors     : Gerard Beekmans - gerard@linuxfromscratch.org
#               DJ Lucas - dj@linuxfromscratch.org
```

```
# Update      : Bruce Dubbs - bdubbs@linuxfromscratch.org
#
# Version     : LFS 7.0
#
########################################################################

### BEGIN INIT INFO
# Provides:            $local_fs
# Required-Start:      udev checkfs
# Should-Start:        modules
# Required-Stop:       localnet
# Should-Stop:
# Default-Start:       S
# Default-Stop:        0 6
# Short-Description:   Mounts/unmounts local filesystems defined in /etc/fstab.
# Description:         Remounts root filesystem read/write and mounts all
#                      remaining local filesystems defined in /etc/fstab on
#                      start.  Remounts root filesystem read-only and unmounts
#                      remaining filesystems on stop.
# X-LFS-Provided-By:   LFS
### END INIT INFO

. /lib/lsb/init-functions

case "${1}" in
   start)
      log_info_msg "Remounting root file system in read-write mode..."
      mount --options remount,rw / >/dev/null
      evaluate_retval

      # Remove fsck-related file system watermarks.
      rm -f /fastboot /forcefsck

      # Make sure /dev/pts exists
      mkdir -p /dev/pts

      # This will mount all filesystems that do not have _netdev in
      # their option list.  _netdev denotes a network filesystem.

      log_info_msg "Mounting remaining file systems..."
      failed=0
      mount --all --test-opts no_netdev >/dev/null || failed=1
      evaluate_retval
      exit $failed
      ;;

   stop)
      # Don't unmount virtual file systems like /run
      log_info_msg "Unmounting all other currently mounted file systems..."
      # Ensure any loop devices are removed
      losetup -D
      umount --all --detach-loop --read-only \
             --types notmpfs,nosysfs,nodevtmpfs,noproc,nodevpts >/dev/null
      evaluate_retval

      # Make sure / is mounted read only (umount bug)
      mount --options remount,ro /

      # Make all LVM volume groups unavailable, if appropriate
      # This fails if swap or / are on an LVM partition
      #if [ -x /sbin/vgchange ]; then /sbin/vgchange -an > /dev/null; fi
      if [ -r /etc/mdadm.conf ]; then
         log_info_msg "Mark arrays as clean..."
         mdadm --wait-clean --scan
         evaluate_retval
      fi
      ;;
```

```
   *)
      echo "Usage: ${0} {start|stop}"
      exit 1
      ;;
esac

# End mountfs
```

D.10. /etc/rc.d/init.d/udev_retry

```
#!/bin/sh
########################################################################
# Begin udev_retry
#
# Description : Udev cold-plugging script (retry)
#
# Authors     : Alexander E. Patrakov
#               DJ Lucas - dj@linuxfromscratch.org
# Update      : Bruce Dubbs - bdubbs@linuxfromscratch.org
#               Bryan Kadzban -
#
# Version     : LFS 7.0
#
########################################################################

### BEGIN INIT INFO
# Provides:            udev_retry
# Required-Start:      udev
# Should-Start:        $local_fs cleanfs
# Required-Stop:
# Should-Stop:
# Default-Start:       S
# Default-Stop:
# Short-Description:   Replays failed uevents and creates additional devices.
# Description:         Replays any failed uevents that were skipped due to
#                      slow hardware initialization, and creates those needed
#                      device nodes
# X-LFS-Provided-By:   LFS
### END INIT INFO

. /lib/lsb/init-functions

case "${1}" in
   start)
      log_info_msg "Retrying failed uevents, if any..."

      rundir=/run/udev
      # From Debian: "copy the rules generated before / was mounted
      # read-write":

      for file in ${rundir}/tmp-rules--*; do
         dest=${file##*tmp-rules--}
         [ "$dest" = '*' ] && break
         cat $file >> /etc/udev/rules.d/$dest
         rm -f $file
      done

      # Re-trigger the uevents that may have failed,
      # in hope they will succeed now
      /bin/sed -e 's/#.*$//' /etc/sysconfig/udev_retry | /bin/grep -v '^$' | \
      while read line ; do
         for subsystem in $line ; do
            /bin/udevadm trigger --subsystem-match=$subsystem --action=add
         done
      done
```

```
      # Now wait for udevd to process the uevents we triggered
      if ! is_true "$OMIT_UDEV_RETRY_SETTLE"; then
         /bin/udevadm settle
      fi

      evaluate_retval
      ;;

   *)
      echo "Usage ${0} {start}"
      exit 1
      ;;
esac

exit 0

# End udev_retry
```

D.11. /etc/rc.d/init.d/cleanfs

```
#!/bin/sh
########################################################################
# Begin cleanfs
#
# Description : Clean file system
#
# Authors     : Gerard Beekmans - gerard@linuxfromscratch.org
#               DJ Lucas - dj@linuxfromscratch.org
# Update      : Bruce Dubbs - bdubbs@linuxfromscratch.org
#
# Version     : LFS 7.0
#
########################################################################

### BEGIN INIT INFO
# Provides:            cleanfs
# Required-Start:      $local_fs
# Should-Start:
# Required-Stop:
# Should-Stop:
# Default-Start:       S
# Default-Stop:
# Short-Description:   Cleans temporary directories early in the boot process.
# Description:         Cleans temporary directories /run, /var/lock, and
#                      optionally, /tmp.  cleanfs also creates /run/utmp
#                      and any files defined in /etc/sysconfig/createfiles.
# X-LFS-Provided-By:   LFS
### END INIT INFO

. /lib/lsb/init-functions

# Function to create files/directory on boot.
create_files()
{
   # Input to file descriptor 9 and output to stdin (redirection)
   exec 9>&0 < /etc/sysconfig/createfiles

   while read name type perm usr grp dtype maj min junk
   do
      # Ignore comments and blank lines.
      case "${name}" in
         ""|\#*) continue ;;
      esac

      # Ignore existing files.
      if [ ! -e "${name}" ]; then
         # Create stuff based on its type.
```

```
         case "${type}" in
            dir)
               mkdir "${name}"
               ;;
            file)
               :> "${name}"
               ;;
            dev)
               case "${dtype}" in
                  char)
                     mknod "${name}" c ${maj} ${min}
                     ;;
                  block)
                     mknod "${name}" b ${maj} ${min}
                     ;;
                  pipe)
                     mknod "${name}" p
                     ;;
                  *)
                     log_warning_msg "\nUnknown device type: ${dtype}"
                     ;;
               esac
               ;;
            *)
               log_warning_msg "\nUnknown type: ${type}"
               continue
               ;;
         esac

         # Set up the permissions, too.
         chown ${usr}:${grp} "${name}"
         chmod ${perm} "${name}"
      fi
   done

   # Close file descriptor 9 (end redirection)
   exec 0>&9 9>&-
   return 0
}

case "${1}" in
   start)
      log_info_msg "Cleaning file systems:"

      if [ "${SKIPTMPCLEAN}" = "" ]; then
         log_info_msg2 " /tmp"
         cd /tmp &&
         find . -xdev -mindepth 1 ! -name lost+found -delete || failed=1
      fi

      > /run/utmp

      if grep -q '^utmp:' /etc/group ; then
         chmod 664 /run/utmp
         chgrp utmp /run/utmp
      fi

      (exit ${failed})
      evaluate_retval

      if grep -E -qv '^(#|$)' /etc/sysconfig/createfiles 2>/dev/null; then
         log_info_msg "Creating files and directories... "
         create_files      # Always returns 0
         evaluate_retval
      fi

      exit $failed
```

```
      ;;
   *)
      echo "Usage: ${0} {start}"
      exit 1
      ;;
esac

# End cleanfs
```

D.12. /etc/rc.d/init.d/console

```
#!/bin/sh
########################################################################
# Begin console
#
# Description : Sets keymap and screen font
#
# Authors     : Gerard Beekmans - gerard@linuxfromscratch.org
#               Alexander E. Patrakov
#               DJ Lucas - dj@linuxfromscratch.org
# Update      : Bruce Dubbs - bdubbs@linuxfromscratch.org
#
# Version     : LFS 7.0
#
########################################################################

### BEGIN INIT INFO
# Provides:            console
# Required-Start:      $local_fs
# Should-Start:        udev_retry
# Required-Stop:
# Should-Stop:
# Default-Start:       S
# Default-Stop:
# Short-Description:   Sets up a localised console.
# Description:         Sets up fonts and language settings for the user's
#                      local as defined by /etc/sysconfig/console.
# X-LFS-Provided-By:   LFS
### END INIT INFO

. /lib/lsb/init-functions

# Native English speakers probably don't have /etc/sysconfig/console at all
[ -r /etc/sysconfig/console ] && . /etc/sysconfig/console

failed=0

case "${1}" in
   start)
      # See if we need to do anything
      if [ -z "${KEYMAP}"         ] && [ -z "${KEYMAP_CORRECTIONS}" ] &&
         [ -z "${FONT}"           ] && [ -z "${LEGACY_CHARSET}"     ] &&
         ! is_true "${UNICODE}"; then
         exit 0
      fi

      # There should be no bogus failures below this line!
      log_info_msg "Setting up Linux console..."

      # Figure out if a framebuffer console is used
      [ -d /sys/class/graphics/fbcon ] && use_fb=1 || use_fb=0

      # Figure out the command to set the console into the
      # desired mode
      is_true "${UNICODE}" &&
         MODE_COMMAND="echo -en '\033%G' && kbd_mode -u" ||
         MODE_COMMAND="echo -en '\033%@\033(K' && kbd_mode -a"
```

```
      # On framebuffer consoles, font has to be set for each vt in
      # UTF-8 mode. This doesn't hurt in non-UTF-8 mode also.

      ! is_true "${use_fb}" || [ -z "${FONT}" ] ||
         MODE_COMMAND="${MODE_COMMAND} && setfont ${FONT}"

      # Apply that command to all consoles mentioned in
      # /etc/inittab. Important: in the UTF-8 mode this should
      # happen before setfont, otherwise a kernel bug will
      # show up and the unicode map of the font will not be
      # used.

      for TTY in `grep '^[^#].*respawn:/sbin/agetty' /etc/inittab |
         grep -o '\btty[[:digit:]]*\b'`
      do
         openvt -f -w -c ${TTY#tty} -- \
            /bin/sh -c "${MODE_COMMAND}" || failed=1
      done

      # Set the font (if not already set above) and the keymap
      [ "${use_fb}" == "1" ] || [ -z "${FONT}" ] || setfont $FONT || failed=1

      [ -z "${KEYMAP}" ] ||
         loadkeys ${KEYMAP} >/dev/null 2>&1 ||
         failed=1

      [ -z "${KEYMAP_CORRECTIONS}" ] ||
         loadkeys ${KEYMAP_CORRECTIONS} >/dev/null 2>&1 ||
         failed=1

      # Convert the keymap from $LEGACY_CHARSET to UTF-8
      [ -z "$LEGACY_CHARSET" ] ||
         dumpkeys -c "$LEGACY_CHARSET" | loadkeys -u >/dev/null 2>&1 ||
         failed=1

      # If any of the commands above failed, the trap at the
      # top would set $failed to 1
      ( exit $failed )
      evaluate_retval

      exit $failed
      ;;

   *)
      echo "Usage:  ${0} {start}"
      exit 1
      ;;
esac

# End console
```

D.13. /etc/rc.d/init.d/localnet

```
#!/bin/sh
########################################################################
# Begin localnet
#
# Description : Loopback device
#
# Authors     : Gerard Beekmans  - gerard@linuxfromscratch.org
#               DJ Lucas - dj@linuxfromscratch.org
# Update      : Bruce Dubbs - bdubbs@linuxfromscratch.org
#
# Version     : LFS 7.0
#
########################################################################
```

```
### BEGIN INIT INFO
# Provides:            localnet
# Required-Start:      mountvirtfs
# Should-Start:        modules
# Required-Stop:
# Should-Stop:
# Default-Start:       S
# Default-Stop:        0 6
# Short-Description:   Starts the local network.
# Description:         Sets the hostname of the machine and starts the
#                      loopback interface.
# X-LFS-Provided-By:   LFS
### END INIT INFO

. /lib/lsb/init-functions
[ -r /etc/sysconfig/network ] && . /etc/sysconfig/network
[ -r /etc/hostname ] && HOSTNAME=`cat /etc/hostname`

case "${1}" in
   start)
      log_info_msg "Bringing up the loopback interface..."
      ip addr add 127.0.0.1/8 label lo dev lo
      ip link set lo up
      evaluate_retval

      log_info_msg "Setting hostname to ${HOSTNAME}..."
      hostname ${HOSTNAME}
      evaluate_retval
      ;;

   stop)
      log_info_msg "Bringing down the loopback interface..."
      ip link set lo down
      evaluate_retval
      ;;

   restart)
      ${0} stop
      sleep 1
      ${0} start
      ;;

   status)
      echo "Hostname is: $(hostname)"
      ip link show lo
      ;;

   *)
      echo "Usage: ${0} {start|stop|restart|status}"
      exit 1
      ;;
esac

exit 0

# End localnet
```

D.14. /etc/rc.d/init.d/sysctl

```
#!/bin/sh
########################################################################
# Begin sysctl
#
# Description : File uses /etc/sysctl.conf to set kernel runtime
#               parameters
#
```

```
# Authors     : Nathan Coulson (nathan@linuxfromscratch.org)
#               Matthew Burgress (matthew@linuxfromscratch.org)
#               DJ Lucas - dj@linuxfromscratch.org
# Update      : Bruce Dubbs - bdubbs@linuxfromscratch.org
#
# Version     : LFS 7.0
#
########################################################################

### BEGIN INIT INFO
# Provides:            sysctl
# Required-Start:      mountvirtfs
# Should-Start:        console
# Required-Stop:
# Should-Stop:
# Default-Start:       S
# Default-Stop:
# Short-Description:   Makes changes to the proc filesystem
# Description:         Makes changes to the proc filesystem as defined in
#                      /etc/sysctl.conf.  See 'man sysctl(8)'.
# X-LFS-Provided-By:   LFS
### END INIT INFO

. /lib/lsb/init-functions

case "${1}" in
   start)
      if [ -f "/etc/sysctl.conf" ]; then
         log_info_msg "Setting kernel runtime parameters..."
         sysctl -q -p
         evaluate_retval
      fi
      ;;

   status)
      sysctl -a
      ;;

   *)
      echo "Usage: ${0} {start|status}"
      exit 1
      ;;
esac

exit 0

# End sysctl
```

D.15. /etc/rc.d/init.d/sysklogd

```
#!/bin/sh
########################################################################
# Begin sysklogd
#
# Description : Sysklogd loader
#
# Authors     : Gerard Beekmans - gerard@linuxfromscratch.org
#               DJ Lucas - dj@linuxfromscratch.org
# Update      : Bruce Dubbs - bdubbs@linuxfromscratch.org
# Update      : Bruce Dubbs - bdubbs@linuxfromscratch.org LFS12.1
#               Remove kernel log daemon.  The functionality has been
#               merged with syslogd.
#
# Version     : LFS 7.0
#
########################################################################
```

```
### BEGIN INIT INFO
# Provides:            $syslog
# Required-Start:      $first localnet
# Should-Start:
# Required-Stop:       $local_fs
# Should-Stop:         sendsignals
# Default-Start:       2 3 4 5
# Default-Stop:        0 1 6
# Short-Description:   Starts system log daemon.
# Description:         Starts system log daemon.
#                      /etc/fstab.
# X-LFS-Provided-By:   LFS
### END INIT INFO

. /lib/lsb/init-functions

case "${1}" in
   start)
      log_info_msg "Starting system log daemon..."
      parms=${SYSKLOGD_PARMS-'-m 0'}
      start_daemon /sbin/syslogd $parms
      evaluate_retval
      ;;

   stop)
      log_info_msg "Stopping system log daemon..."
      killproc /sbin/syslogd
      evaluate_retval
      ;;

   reload)
      log_info_msg "Reloading system log daemon config file..."
      pid=`pidofproc syslogd`
      kill -HUP "${pid}"
      evaluate_retval
      ;;

   restart)
      ${0} stop
      sleep 1
      ${0} start
      ;;

   status)
      statusproc /sbin/syslogd
      ;;

   *)
      echo "Usage: ${0} {start|stop|reload|restart|status}"
      exit 1
      ;;
esac

exit 0

# End sysklogd
```

D.16. /etc/rc.d/init.d/network

```
#!/bin/sh
########################################################################
# Begin network
#
# Description : Network Control Script
#
# Authors     : Gerard Beekmans - gerard@linuxfromscratch.org
#               Nathan Coulson - nathan@linuxfromscratch.org
```

```
#               Kevin P. Fleming - kpfleming@linuxfromscratch.org
#               DJ Lucas - dj@linuxfromscratch.org
# Update      : Bruce Dubbs - bdubbs@linuxfromscratch.org
#
# Version     : LFS 7.0
#
########################################################################

### BEGIN INIT INFO
# Provides:            $network
# Required-Start:      $local_fs localnet swap
# Should-Start:        $syslog firewalld iptables nftables
# Required-Stop:       $local_fs localnet swap
# Should-Stop:         $syslog firewalld iptables nftables
# Default-Start:       2 3 4 5
# Default-Stop:        0 1 6
# Short-Description:   Starts and configures network interfaces.
# Description:         Starts and configures network interfaces.
# X-LFS-Provided-By:   LFS
### END INIT INFO

case "${1}" in
   start)
      # if the default route exists, network is already configured
      if ip route | grep -q "^default"; then return 0; fi
      # Start all network interfaces
      for file in /etc/sysconfig/ifconfig.*
      do
         interface=${file##*/ifconfig.}

         # Skip if $file is * (because nothing was found)
         if [ "${interface}" = "*" ]; then continue; fi

         /sbin/ifup ${interface}
      done
      ;;

   stop)
      # Unmount any network mounted file systems
       umount --all --force --types nfs,cifs,nfs4

      # Reverse list
      net_files=""
      for file in  /etc/sysconfig/ifconfig.*
      do
         net_files="${file} ${net_files}"
      done

      # Stop all network interfaces
      for file in ${net_files}
      do
         interface=${file##*/ifconfig.}

         # Skip if $file is * (because nothing was found)
         if [ "${interface}" = "*" ]; then continue; fi

         # See if interface exists
         if [ ! -e /sys/class/net/$interface ]; then continue; fi

         # Is interface UP?
         ip link show $interface 2>/dev/null | grep -q "state UP"
         if [ $? -ne 0 ];  then continue; fi

         /sbin/ifdown ${interface}
      done
      ;;
```

```
   restart)
      ${0} stop
      sleep 1
      ${0} start
      ;;

   *)
      echo "Usage: ${0} {start|stop|restart}"
      exit 1
      ;;
esac

exit 0

# End network
```

D.17. /etc/rc.d/init.d/sendsignals

```
#!/bin/sh
########################################################################
# Begin sendsignals
#
# Description : Sendsignals Script
#
# Authors     : Gerard Beekmans - gerard@linuxfromscratch.org
#               DJ Lucas - dj@linuxfromscratch.org
# Update      : Bruce Dubbs - bdubbs@linuxfromscratch.org
#
# Version     : LFS 7.0
#
########################################################################

### BEGIN INIT INFO
# Provides:            sendsignals
# Required-Start:
# Should-Start:
# Required-Stop:       $local_fs swap localnet
# Should-Stop:
# Default-Start:
# Default-Stop:        0 6
# Short-Description:   Attempts to kill remaining processes.
# Description:         Attempts to kill remaining processes.
# X-LFS-Provided-By:   LFS
### END INIT INFO

. /lib/lsb/init-functions

case "${1}" in
   stop)
      omit=$(pidof mdmon)
      [ -n "$omit" ] && omit="-o $omit"

      log_info_msg "Sending all processes the TERM signal..."
      killall5 -15 $omit
      error_value=${?}

      sleep ${KILLDELAY}

      if [ "${error_value}" = 0 -o "${error_value}" = 2 ]; then
         log_success_msg
      else
         log_failure_msg
      fi

      log_info_msg "Sending all processes the KILL signal..."
      killall5 -9 $omit
      error_value=${?}
```

```
      sleep ${KILLDELAY}

      if [ "${error_value}" = 0 -o "${error_value}" = 2 ]; then
         log_success_msg
      else
         log_failure_msg
      fi
      ;;

   *)
      echo "Usage: ${0} {stop}"
      exit 1
      ;;

esac

exit 0

# End sendsignals
```

D.18. /etc/rc.d/init.d/reboot

```
#!/bin/sh
###############################################################################
# Begin reboot
#
# Description : Reboot Scripts
#
# Authors     : Gerard Beekmans - gerard@linuxfromscratch.org
#               DJ Lucas - dj@linuxfromscratch.org
# Updates     : Bruce Dubbs - bdubbs@linuxfromscratch.org
#             : Pierre Labastie - pierre@linuxfromscratch.org
#
# Version     : LFS 7.0
#
# Notes       : Update March 24th, 2022: change "stop" to "start".
#               Add the $last facility to Required-start
#
###############################################################################

### BEGIN INIT INFO
# Provides:            reboot
# Required-Start:      $last
# Should-Start:
# Required-Stop:
# Should-Stop:
# Default-Start:       6
# Default-Stop:
# Short-Description:   Reboots the system.
# Description:         Reboots the System.
# X-LFS-Provided-By:   LFS
### END INIT INFO

. /lib/lsb/init-functions

case "${1}" in
   start)
      log_info_msg "Restarting system..."
      reboot -d -f -i
      ;;

   *)
      echo "Usage: ${0} {start}"
      exit 1
      ;;
```

```
esac

# End reboot
```

D.19. /etc/rc.d/init.d/halt

```
#!/bin/sh
########################################################################
# Begin halt
#
# Description : Halt Script
#
# Authors     : Gerard Beekmans - gerard@linuxfromscratch.org
#               DJ Lucas - dj@linuxfromscratch.org
# Update      : Bruce Dubbs - bdubbs@linuxfromscratch.org
#             : Pierre Labastie - pierre@linuxfromscratch.org
#
# Version     : LFS 7.0
#
# Notes       : Update March 24th, 2022: change "stop" to "start".
#               Add the $last facility to Required-start
#
########################################################################

### BEGIN INIT INFO
# Provides:            halt
# Required-Start:      $last
# Should-Start:
# Required-Stop:
# Should-Stop:
# Default-Start:       0
# Default-Stop:
# Short-Description:   Halts the system.
# Description:         Halts the System.
# X-LFS-Provided-By:   LFS
### END INIT INFO

case "${1}" in
   start)
      halt -d -f -i -p
      ;;

   *)
      echo "Usage: {start}"
      exit 1
      ;;
esac

# End halt
```

D.20. /etc/rc.d/init.d/template

```
#!/bin/sh
########################################################################
# Begin scriptname
#
# Description :
#
# Authors     :
#
# Version     : LFS x.x
#
# Notes       :
#
########################################################################
```

```
### BEGIN INIT INFO
# Provides:            template
# Required-Start:
# Should-Start:
# Required-Stop:
# Should-Stop:
# Default-Start:
# Default-Stop:
# Short-Description:
# Description:
# X-LFS-Provided-By:
### END INIT INFO

. /lib/lsb/init-functions

case "${1}" in
   start)
      log_info_msg "Starting..."
    # if it is possible to use start_daemon
      start_daemon fully_qualified_path
    # if it is not possible to use start_daemon
    # (command to start the daemon is not simple enough)
      if ! pidofproc daemon_name_as_reported_by_ps >/dev/null; then
         command_to_start_the_service
      fi
      evaluate_retval
      ;;

   stop)
      log_info_msg "Stopping..."
    # if it is possible to use killproc
      killproc fully_qualified_path
    # if it is not possible to use killproc
    # (the daemon shouldn't be stopped by killing it)
      if pidofproc daemon_name_as_reported_by_ps >/dev/null; then
         command_to_stop_the_service
      fi
      evaluate_retval
      ;;

   restart)
      ${0} stop
      sleep 1
      ${0} start
      ;;

   *)
      echo "Usage: ${0} {start|stop|restart}"
      exit 1
      ;;
esac

exit 0

# End scriptname
```

D.21. /etc/sysconfig/modules

```
#############################################################################
# Begin /etc/sysconfig/modules
#
# Description : Module auto-loading configuration
#
# Authors     :
#
# Version     : 00.00
#
```

```
# Notes       : The syntax of this file is as follows:
#               <module> [<arg1> <arg2> ...]
#
# Each module should be on its own line, and any options that you want
# passed to the module should follow it.  The line deliminator is either
# a space or a tab.
#########################################################################

# End /etc/sysconfig/modules
```

D.22. /etc/sysconfig/createfiles

```
#########################################################################
# Begin /etc/sysconfig/createfiles
#
# Description : Createfiles script config file
#
# Authors     :
#
# Version     : 00.00
#
# Notes       : The syntax of this file is as follows:
#               if type is equal to "file" or "dir"
#                <filename> <type> <permissions> <user> <group>
#               if type is equal to "dev"
#                <filename> <type> <permissions> <user> <group> <devtype>
#             <major> <minor>
#
#               <filename> is the name of the file which is to be created
#               <type> is either file, dir, or dev.
#                       file creates a new file
#                       dir creates a new directory
#                       dev creates a new device
#               <devtype> is either block, char or pipe
#                       block creates a block device
#                       char creates a character device
#                       pipe creates a pipe, this will ignore the <major> and
#           <minor> fields
#                <major> and <minor> are the major and minor numbers used for
#     the device.
#########################################################################

# End /etc/sysconfig/createfiles
```

D.23. /etc/sysconfig/udev-retry

```
#########################################################################
# Begin /etc/sysconfig/udev_retry
#
# Description : udev_retry script configuration
#
# Authors     :
#
# Version     : 00.00
#
# Notes       : Each subsystem that may need to be re-triggered after mountfs
#               runs should be listed in this file.  Probable subsystems to be
#               listed here are rtc (due to /var/lib/hwclock/adjtime) and sound
#               (due to both /var/lib/alsa/asound.state and /usr/sbin/alsactl).
#               Entries are whitespace-separated.
#########################################################################

rtc

# End /etc/sysconfig/udev_retry
```

D.24. /sbin/ifup

```
#!/bin/sh
########################################################################
# Begin /sbin/ifup
#
# Description : Interface Up
#
# Authors     : Nathan Coulson - nathan@linuxfromscratch.org
#               Kevin P. Fleming - kpfleming@linuxfromscratch.org
# Update      : Bruce Dubbs - bdubbs@linuxfromscratch.org
#               DJ Lucas - dj@linuxfromscratch.org
#
# Version     : LFS 7.7
#
# Notes       : The IFCONFIG variable is passed to the SERVICE script
#               in the /lib/services directory, to indicate what file the
#               service should source to get interface specifications.
#
########################################################################

up()
{
  log_info_msg "Bringing up the ${1} interface..."

  if ip link show $1 > /dev/null 2>&1; then
     link_status=`ip link show $1`

     if [ -n "${link_status}" ]; then
        if ! echo "${link_status}" | grep -q UP; then
           ip link set $1 up
        fi
     fi

  else
     log_failure_msg "Interface ${IFACE} doesn't exist."
     exit 1
  fi

  evaluate_retval
}

RELEASE="7.7"

USAGE="Usage: $0 [ -hV ] [--help] [--version] interface"
VERSTR="LFS ifup, version ${RELEASE}"

while [ $# -gt 0 ]; do
   case "$1" in
      --help | -h)     help="y"; break ;;

      --version | -V)  echo "${VERSTR}"; exit 0 ;;

      -*)              echo "ifup: ${1}: invalid option" >&2
                       echo "${USAGE}" >& 2
                       exit 2 ;;

      *)               break ;;
   esac
done

if [ -n "$help" ]; then
   echo "${VERSTR}"
   echo "${USAGE}"
   echo
   cat << HERE_EOF
ifup is used to bring up a network interface.  The interface
```

```
parameter, e.g. eth0 or eth0:2, must match the trailing part of the
interface specifications file, e.g. /etc/sysconfig/ifconfig.eth0:2.

HERE_EOF
   exit 0
fi

file=/etc/sysconfig/ifconfig.${1}

# Skip backup files
[ "${file}" = "${file%""~""}" ] || exit 0

. /lib/lsb/init-functions

if [ ! -r "${file}" ]; then
   log_failure_msg "Unable to bring up ${1} interface! ${file} is missing or cannot be accessed."
   exit 1
fi

.  $file

if [ "$IFACE" = "" ]; then
   log_failure_msg "Unable to bring up ${1} interface! ${file} does not define an interface [IFACE]."
   exit 1
fi

# Do not process this service if started by boot, and ONBOOT
# is not set to yes
if [ "${IN_BOOT}" = "1" -a "${ONBOOT}" != "yes" ]; then
   exit 0
fi

# Bring up the interface
if [ "$VIRTINT" != "yes" ]; then
   up ${IFACE}
fi

for S in ${SERVICE}; do
  if [ ! -x "/lib/services/${S}" ]; then
    MSG="\nUnable to process ${file}.  Either "
    MSG="${MSG}the SERVICE '${S} was not present "
    MSG="${MSG}or cannot be executed."
    log_failure_msg "$MSG"
    exit 1
  fi
done

#if [ "${SERVICE}" = "wpa" ]; then log_success_msg; fi

# Create/configure the interface
for S in ${SERVICE}; do
  IFCONFIG=${file} /lib/services/${S} ${IFACE} up
done

# Set link up virtual interfaces
if [ "${VIRTINT}" == "yes" ]; then
   up ${IFACE}
fi

# Bring up any additional interface components
for I in $INTERFACE_COMPONENTS; do up $I; done

# Set MTU if requested. Check if MTU has a "good" value.
if test -n "${MTU}"; then
   if [[ ${MTU} =~ ^[0-9]+$ ]] && [[ $MTU -ge 68 ]] ; then
      for I in $IFACE $INTERFACE_COMPONENTS; do
         ip link set dev $I mtu $MTU;
```

```
      done
   else
      log_info_msg2 "Invalid MTU $MTU"
   fi
fi

# Set the route default gateway if requested
if [ -n "${GATEWAY}" ]; then
   if ip route | grep -q default; then
      log_warning_msg "Gateway already setup; skipping."
   else
      log_info_msg "Adding default gateway ${GATEWAY} to the ${IFACE} interface..."
      ip route add default via ${GATEWAY} dev ${IFACE}
      evaluate_retval
   fi
fi

# End /sbin/ifup
```

D.25. /sbin/ifdown

```
#!/bin/bash
########################################################################
# Begin /sbin/ifdown
#
# Description : Interface Down
#
# Authors     : Nathan Coulson - nathan@linuxfromscratch.org
#               Kevin P. Fleming - kpfleming@linuxfromscratch.org
# Update      : Bruce Dubbs - bdubbs@linuxfromscratch.org
#
# Version     : LFS 7.0
#
# Notes       : the IFCONFIG variable is passed to the scripts found
#               in the /lib/services directory, to indicate what file the
#               service should source to get interface specifications.
#
########################################################################

RELEASE="7.0"

USAGE="Usage: $0 [ -hV ] [--help] [--version] interface"
VERSTR="LFS ifdown, version ${RELEASE}"

while [ $# -gt 0 ]; do
   case "$1" in
      --help | -h)     help="y"; break ;;

      --version | -V)  echo "${VERSTR}"; exit 0 ;;

      -*)              echo "ifup: ${1}: invalid option" >&2
                       echo "${USAGE}" >& 2
                       exit 2 ;;

      *)               break ;;
   esac
done

if [ -n "$help" ]; then
   echo "${VERSTR}"
   echo "${USAGE}"
   echo
   cat << HERE_EOF
ifdown is used to bring down a network interface.  The interface
parameter, e.g. eth0 or eth0:2, must match the trailing part of the
interface specifications file, e.g. /etc/sysconfig/ifconfig.eth0:2.
```

```
HERE_EOF
   exit 0
fi

file=/etc/sysconfig/ifconfig.${1}

# Skip backup files
[ "${file}" = "${file%""~""}" ] || exit 0

. /lib/lsb/init-functions

if [ ! -r "${file}" ]; then
   log_warning_msg "${file} is missing or cannot be accessed."
   exit 1
fi

. ${file}

if [ "$IFACE" = "" ]; then
   log_failure_msg "${file} does not define an interface [IFACE]."
   exit 1
fi

# We only need to first service to bring down the interface
S=`echo ${SERVICE} | cut -f1 -d" "`

if ip link show ${IFACE} > /dev/null 2>&1; then
   if [ -n "${S}" -a -x "/lib/services/${S}" ]; then
     IFCONFIG=${file} /lib/services/${S} ${IFACE} down
   else
     MSG="Unable to process ${file}.  Either "
     MSG="${MSG}the SERVICE variable was not set "
     MSG="${MSG}or the specified service cannot be executed."
     log_failure_msg "$MSG"
     exit 1
  fi
else
   log_warning_msg "Interface ${1} doesn't exist."
fi

# Leave the interface up if there are additional interfaces in the device
link_status=`ip link show ${IFACE} 2>/dev/null`

if [ -n "${link_status}" ]; then
   if [ "$(echo "${link_status}" | grep UP)" != "" ]; then
      if [ "$(ip addr show ${IFACE} | grep 'inet ')" == ""  ]; then
         log_info_msg "Bringing down the ${IFACE} interface..."
         ip link set ${IFACE} down
         evaluate_retval
      fi
   fi
fi

# End /sbin/ifdown
```

D.26. /lib/services/ipv4-static

```
#!/bin/sh
########################################################################
# Begin /lib/services/ipv4-static
#
# Description : IPV4 Static Boot Script
#
# Authors     : Nathan Coulson - nathan@linuxfromscratch.org
#               Kevin P. Fleming - kpfleming@linuxfromscratch.org
# Update      : Bruce Dubbs - bdubbs@linuxfromscratch.org
#
```

```
# Version     : LFS 7.0
#
########################################################################

. /lib/lsb/init-functions
. ${IFCONFIG}

if [ -z "${IP}" ]; then
   log_failure_msg "\nIP variable missing from ${IFCONFIG}, cannot continue."
   exit 1
fi

if [ -z "${PREFIX}" -a -z "${PEER}" ]; then
   log_warning_msg "\nPREFIX variable missing from ${IFCONFIG}, assuming 24."
   PREFIX=24
   args="${args} ${IP}/${PREFIX}"

elif [ -n "${PREFIX}" -a -n "${PEER}" ]; then
   log_failure_msg "\nPREFIX and PEER both specified in ${IFCONFIG}, cannot continue."
   exit 1

elif [ -n "${PREFIX}" ]; then
   args="${args} ${IP}/${PREFIX}"

elif [ -n "${PEER}" ]; then
   args="${args} ${IP} peer ${PEER}"
fi

if [ -n "${LABEL}" ]; then
   args="${args} label ${LABEL}"
fi

if [ -n "${BROADCAST}" ]; then
   args="${args} broadcast ${BROADCAST}"
fi

case "${2}" in
   up)
      if [ "$(ip addr show ${1} 2>/dev/null | grep ${IP}/)" = "" ]; then
         log_info_msg "Adding IPv4 address ${IP} to the ${1} interface..."
         ip addr add ${args} dev ${1}
         evaluate_retval
      else
         log_warning_msg "Cannot add IPv4 address ${IP} to ${1}.  Already present."
      fi
   ;;

   down)
      if [ "$(ip addr show ${1} 2>/dev/null | grep ${IP}/)" != "" ]; then
         log_info_msg "Removing IPv4 address ${IP} from the ${1} interface..."
         ip addr del ${args} dev ${1}
         evaluate_retval
      fi

      if [ -n "${GATEWAY}" ]; then
         # Only remove the gateway if there are no remaining ipv4 addresses
         if [ "$(ip addr show ${1} 2>/dev/null | grep 'inet ')" != "" ]; then
            log_info_msg "Removing default gateway..."
            ip route del default
            evaluate_retval
         fi
      fi
   ;;

   *)
      echo "Usage: ${0} [interface] {up|down}"
      exit 1
```

```
   ;;
esac

# End /lib/services/ipv4-static
```

D.27. /lib/services/ipv4-static-route

```
#!/bin/sh
########################################################################
# Begin /lib/services/ipv4-static-route
#
# Description : IPV4 Static Route Script
#
# Authors     : Kevin P. Fleming - kpfleming@linuxfromscratch.org
#               DJ Lucas - dj@linuxfromscratch.org
# Update      : Bruce Dubbs - bdubbs@linuxfromscratch.org
#
# Version     : LFS 7.0
#
########################################################################

. /lib/lsb/init-functions
. ${IFCONFIG}

case "${TYPE}" in
   ("" | "network")
      need_ip=1
      need_gateway=1
   ;;

   ("default")
      need_gateway=1
      args="${args} default"
      desc="default"
   ;;

   ("host")
      need_ip=1
   ;;

   ("unreachable")
      need_ip=1
      args="${args} unreachable"
      desc="unreachable "
   ;;

   (*)
      log_failure_msg "Unknown route type (${TYPE}) in ${IFCONFIG}, cannot continue."
      exit 1
   ;;
esac

if [ -n "${GATEWAY}" ]; then
   MSG="The GATEWAY variable cannot be set in ${IFCONFIG} for static routes.\n"
   log_failure_msg "$MSG Use STATIC_GATEWAY only, cannot continue"
   exit 1
fi

if [ -n "${need_ip}" ]; then
   if [ -z "${IP}" ]; then
      log_failure_msg "IP variable missing from ${IFCONFIG}, cannot continue."
      exit 1
   fi

   if [ -z "${PREFIX}" ]; then
      log_failure_msg "PREFIX variable missing from ${IFCONFIG}, cannot continue."
      exit 1
```

```
   fi

   args="${args} ${IP}/${PREFIX}"
   desc="${desc}${IP}/${PREFIX}"
fi

if [ -n "${need_gateway}" ]; then
   if [ -z "${STATIC_GATEWAY}" ]; then
      log_failure_msg "STATIC_GATEWAY variable missing from ${IFCONFIG}, cannot continue."
      exit 1
   fi
   args="${args} via ${STATIC_GATEWAY}"
fi

if [ -n "${SOURCE}" ]; then
        args="${args} src ${SOURCE}"
fi

case "${2}" in
   up)
      log_info_msg "Adding '${desc}' route to the ${1} interface..."
      ip route add ${args} dev ${1}
      evaluate_retval
   ;;

   down)
      log_info_msg "Removing '${desc}' route from the ${1} interface..."
      ip route del ${args} dev ${1}
      evaluate_retval
   ;;

   *)
      echo "Usage: ${0} [interface] {up|down}"
      exit 1
   ;;
esac

# End /lib/services/ipv4-static-route
```

Приложение E. Правила настройки Udev

Правила в этом приложении перечислены для удобства. Установка обычно выполняются с помощью инструкций приведенных в Раздел 8.77, «Udev из Systemd-256.4».

E.1. 55-lfs.rules

```
# /etc/udev/rules.d/55-lfs.rules: Rule definitions for LFS.

# Core kernel devices

# This causes the system clock to be set as soon as /dev/rtc becomes available.
SUBSYSTEM=="rtc", ACTION=="add", MODE="0644", RUN+="/etc/rc.d/init.d/setclock start"
KERNEL=="rtc", ACTION=="add", MODE="0644", RUN+="/etc/rc.d/init.d/setclock start"
```

Приложение F. Лицензии LFS

F.1. Лицензия Creative Commons

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.

Важно

Creative Commons is not a party to this License, and makes no warranty whatsoever in connection with the Work. Creative Commons will not be liable to You or any party on any legal theory for any damages whatsoever, including without limitation any general, special, incidental or consequential damages arising in connection to this license. Notwithstanding the foregoing two (2) sentences, if Creative Commons has expressly identified itself as the Licensor hereunder, it shall have all rights and obligations of Licensor.

Except for the limited purpose of indicating to the public that the Work is licensed under the CCPL, neither party will use the trademark "Creative Commons" or any related trademark or logo of Creative Commons without the prior written consent of Creative Commons. Any permitted use will be in compliance with Creative Commons' then-current trademark usage guidelines, as may be published on its website or otherwise made available upon request from time to time.

Creative Commons may be contacted at *http://creativecommons.org/*.

F.2. Лицензия MIT

Предметный указатель

Пакеты

Программы

Библиотеки

Скрипты

Разное

/etc/os-release: 317
/etc/passwd: 103
/etc/profile: 301
/etc/protocols: 123
/etc/resolv.conf: 290
/etc/services: 123
/etc/syslog.conf: 275
/etc/udev: 257, 259
/etc/udev/hwdb.bin: 259
/etc/vimrc: 253
/run/utmp: 103
/usr/include/asm-generic/*.h: 73, 73
/usr/include/asm/*.h: 73, 73
/usr/include/drm/*.h: 73, 73
/usr/include/linux/*.h: 73, 73
/usr/include/misc/*.h: 73, 73
/usr/include/mtd/*.h: 73, 73
/usr/include/rdma/*.h: 73, 73
/usr/include/scsi/*.h: 73, 73
/usr/include/sound/*.h: 73, 73
/usr/include/video/*.h: 73, 73
/usr/include/xen/*.h: 73, 73
/var/log/btmp: 103
/var/log/lastlog: 103
/var/log/wtmp: 103
/etc/shells: 304
man pages: 122, 122