Linux From Scratch

Версия 9.0

Дата публикации 1 Сентября, 2019

Создатель - Gerard Beekmans
Главный редактор: Bruce Dubbs
Технический перевод: Антон Майсак
Редактор перевода: Елена Шевцова

Linux From Scratch: Версия 9.0 : Дата публикации 1 Сентября, 2019

by Создатель - Gerard Beekmans, Главный редактор: Bruce Dubbs, Технический перевод: Антон Майсак, and Редактор перевода: Елена Шевцова

Table of Contents

Предисловие

Предисловие

Моё путешествие в мир Linux началось в 1998 году. Тогда я только установил мой первый Linux-дистрибутив и сразу заинтересовался его концепцией и философией.

У задачи может быть несколько вариантов решения. О Linux-системах можно сказать также. Многие из них существовали годами. Какие-то из них всё еще существуют, какие-то превратились во что-то иное, а некоторые остались только в наших воспоминаниях. Все эти дистрибутивы выполняют задачи по-разному, чтобы удовлетворить потребности своей целевой аудитории. И я осознал - раз существует так много всевозможных способов добиться поставленной цели, мне больше не нужно ограничивать себя в средствах её достижения. До открытия Linux мы часто сталкивались с "багами" других операционных систем, но выбирать не приходилось. 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 даже поощряется.

Я попробовал разные дистрибутивы, но так и не смог ни на одном остановиться. Они были отличными системами сами по себе. Дело даже не в том, хороши они были или плохи - это дело вкуса. При всём разнообразии выбора не было ни одного дистрибутива, который был для меня идеален. Поэтому я решил создать свою собственную Linux-систему, которая полностью соответствовала моим личным предпочтениям.

Чтобы создать свою собственную систему, я решил скомпилировать всё из исходного кода вместо использования ранее 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.

После решения таких проблем, как циклические зависимости и ошибки во время компиляции, я наконец собрал собственную Linux систему. Она была полностью работоспособна и отлично справлялась со своими задачами, как и любая другая ОС с ядром Linux. Но это было моё собственное творение. Было очень приятно собрать такую систему самостоятельно. Лучше этого могло быть только самому создавать каждый компонент системы. Это было следующее, к чему стремиться.

Когда я поделился своими идеями с другими членами сообщества Linux, стал очевиден явный интерес к ним. Скоро стало понятно, что самостоятельно собранные Linux системы готовы были не только соответствовать специфическим потребностям пользователя, но и были идеальны для обучения программистов и системных администраторов с целью повышения их навыков. Из этих интересов и родился проект *Linux From Scratch*.

Книга Linux From Scratch - это ядро всего проекта. Она обеспечит вас знаниями и инструкциями, необходимыми для создания собственой Linux-системы. Эта книга предоставляет шаблон, выполнив который, вы получите правильно работающую систему. С другой стороны, вы имеете возможность менять инструкции на своё усмотрение, что также немаловажно. Вы всё контролируете. Мы просто протягиваем вам руку помощи, чтобы вы смогли начать собственное путешествие.

Я искренне надеюсь, что вам понравится работать над своей собственной системой Linux From Scratch и вы оцените её преимущества.

--
Gerard Beekmans
gerard@linuxfromscratch.org

Аудитория, на которую рассчитана эта книга

Есть много причин, по которым вы хотели бы прочитать эту книгу. Один из вопросов, который задают многие пользователи - зачем проходить через все трудности ручной сборки Linux с нуля, когда можно просто загрузить и установить существующий дистрибутив?

Одна из главных целей существования этого проекта - дать понимание того, как работает Linux изнутри. Создание системы LFS поможет узнать, как всё работает вместе и как каждый компонент системы взаимодействует с другими. Немаловажно, что книга даёт опыт в обучении, и, как следствие, возможность настроить систему Linux в соответствии с потребностями.

Другое ключевое преимущество - LFS предоставляет более глубокий контроль, не полагаясь на другую реализацию Linux. Вы можете диктовать каждый аспект своей системы.

LFS позволяет создать очень компактную Linux систему. Когда вы устанавливаете какой-нибудь дистрибутив, вам принудительно поставят большое количество программ, которыми, скорее всего, вы никогда не воспользуетесь. Они будут только расходовать системные ресурсы. Безусловно, это спорное замечание, так как современное аппаратное обеспечение имеет достаточно высокую производительность и позволяет не беспокоиться о нехватке ресурсов. Но если если рассматривать работу LFS во встраиваемых системах (банкоматы, телекоммуникационное оборудование и т.п.), выгода станет очевидна.

Ещё одним преимуществом собственной сборки Linux является безопасность. При компиляции каждого компонента системы из исходных кодов вы можете всё проверить и применить необходимые патчи. Теперь не надо ждать, когда кто-нибудь 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 проекта Linux From Scratch - собрать окончательную и удобную систему базового уровня. Если вы не хотите создавать собственную Linux-систему с нуля, информация в этой книге может быть в любом случае полезна.

Есть ещё масса причин, по которым следует выполнять сборку собственной системы LFS. В конечном счёте стоит образовательная цель, безусловно, являющаяся самой важной. По мере работы с LFS вы обнаружите силу, которую приносят знания и информация.

Целевые архитектуры LFS

Основными целевыми архитектурами LFS являются AMD / Intel x86 (32-разрядная) и x86_64 (64-разрядная). С другой стороны, инструкции в этой книге также работают, с некоторыми модификациями, с Power PC и ARM процессорами. Чтобы собрать систему, которая должна работать на одном из вышеуказанных процессоров, главным условием является существующая операционная система Linux, например уже ранее собранная LFS, Ubuntu, Red Hat/Fedora, SuSE,

или другой дистрибутив, который нацелен на требуемую архитектуру. Также обратите внимание, что 32-разрядный дистрибутив может быть установлен и использоваться как хост-система на 64-разрядном AMD / Intel компьютере.

Также, необходимо отметить некоторые факты о 64-разрядных системах. По сравнению с 32-разрядной системой, размеры исполняемых программ немного больше, а скорость выполнения - немного быстрее. Например, в тестовой сборке LFS-6.5 на базе процессора Intel Core2Duo были выведены следующие статистические данные:

```
Архитектура Время сборки     Размер сборки
32-bit       198.5 минут  648 MB
64-bit       190.6 минут  709 MB
```

Как видите, 64-разрядная система была собрана на 4% быстрее, и на 19% большего размера, чем 32-разрядная версия. Преимущества 64-разрядной системы относительно небольшое. Безусловно, если у Вас более чем 4GB оперативной памяти, и вы будете работать с такими данными, которые превышают этот порог, то преимущества 64-разрядной системы существенны.

Note

Обсуждение выше применимо, если выполнять сборку на одном и том же аппаратном обеспечении. Современные 64-разрядные системы значительно быстрее, чем старые 64-битные системы, и авторы книги рекомендуют выбирать 64-разрядную машину для сборки.

По умолчанию 64-разрядная сборка LFS, считается "чистой" 64-разрядной системой. То есть она поддерживает только 64-разрядные исполняемые файлы. Сборка "multi-lib" 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. Вы можете перейти по ссылке проекта *Cross Linux From Scratch* для изучения этого вопроса.

LFS and стандарты

Структура LFS следует стандартам Linux на столько, на сколько это возможно:

- *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. Кроме того, существуют требования специфичные для архитектуры. Есть также две области для пробного или ознакомительного использования: Gtk3 и Graphics. LFS старается соответствовать стандартам предусмотренными архитектурами, рассмотренными в предыдущем разделе.

Note

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 идёт из исходных кодов, у пользователя имеется полный контроль над тем, какие пакеты ему необходимы и многие предпочитают не устанавливать некоторые пакеты, которые определяются в стандартах LSB.

Создание окончательной системы LFS, способной успешно выполнять сертификационные тесты LSB - возможно с установкой некоторых дополнительных пакетов, которые выходят за рамки этой книги. Но инструкции по их установке есть в книге BLFS.

Пакеты LFS которые необходимы для следования требованиям 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
LSB Imaging:	Нет
LSB Gtk3 and LSB Graphics (Trial Use):	Нет

Пакеты BLFS которые необходимы для следования требованиям LSB

LSB Core:	At, Batch (a part of At), Cpio, Ed, Fcrontab, Initd-tools, Lsb_release, NSPR, NSS, PAM, Pax, Sendmail (or Postfix or 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:	Python, Libxml2, Libxslt
LSB Imaging:	CUPS, Cups-filters, Ghostscript, SANE
LSB Gtk3 and LSB Graphics (Trial Use):	GTK+3

Пакеты которые отсутствуют в LFS и BLFS, которые необходимы для следования требованиям LSB

LSB Core:	Нет
LSB Desktop:	Qt4 (но Qt5 предоставляется)
LSB Runtime Languages:	Нет
LSB Imaging:	Нет

LSB Gtk3 and LSB Graphics (Trial Use): Нет

Информация о пакетах, используемых в этой книге

Как говорилось ранее, целью проекта Linux From Scratch является сборка системы базового уровня. Она будет включать в себя пакеты необходимые для репликации и распространения, а также относительно небольшой набор программ, с помощью которых можно расширять систему в любом направлении на ваше усмотрение. Это не значит, что LFS является максимально компактной. Есть пакеты, которые включены но строго не требуются. В списке, который расположен ниже, имеются описания для каждого пакета.

- 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 списками управления доступом (ACL).

- Attr

  Программы для администрирования расширенных атрибутов объектов файловой системы.

- Autoconf

  Программы для воспроизведения сценариев командной оболочки которые могут выполнять автоматическую настройку исходного кода из определенного пользовательского файла-шаблона. Он также необходим для повторной компиляции пакета после обновления процедур сборки.

- Automake

  Программы для создания файлов Makefile для использования его программой Autoconf. Он также необходим для повторной компиляции пакета после обновления процедур сборки.

- Bash

  Усовершенствованная и модернизированная вариация командной оболочки Bourne shell. Этот пакет выполняет требования стандарта LFS Core для обеспечения интерфейса Bourne Shell в системе. Он был выбран из числа других оболочек из-за широкого распространения, возможностей которые выходят далеко за пределы базовых функций программ-оболочек.

- Bc

  Произвольный язык обработки числовой точности. Он необходим для сборки Linux ядра.

- Binutils

  компоновщик, ассемблер, и другие утилиты и инструменты для работы с объектными файлам. Программы в этом пакете необходимы для компиляции как большинства пакетов системы LFS, так и многих пакетов за её пределами.

- Bison

  GNU версия yacc (Ещё один компилятор компиляторов) необходимый для сборки некоторых пакетов для системы LFS.

- Bzip2

Программы для сжатия и распаковки файлов. Он необходим для распаковки многих пакетов LFS.

- Check

  Средства тестирования для других программ. Он будет установлен только во временный инструментарий.

- Coreutils

  Программы для просмотра и манипулирования файлами и каталогами. Эти программы необходимы для управления файлами через командную строку, и необходимы для процедуры установки каждого пакета в LFS.

- DejaGNU

  Фреймворк для тестирования других программ. Он будет установлен только во временный инструментарий.

- Diffutils

  Программы, которые отображают разницу в содержимом между файлами и каталогами. Эти программы могут быть использованы, для создания патчей, а также они используются в процедурах сборки для большинства пакетов.

- E2fsprogs

  Утилиты для обработки файловых систем ext2, ext3 и ext4. Это наиболее распространенные и тщательно протестированные файловые системы, поддерживаемые Linux.

- Eudev

  Диспетчер устройств. Он контролирует записи в каталоге /dev, так как устройства добавляются или удаляются из системы динамически.

- Expat

  Небольшая библиотека для обработки (парсинга) XML. Она необходима для пакета модуля Perl - XML::Parser.

- Expect

  Инструменты для автоматизации и тестирования, и является расширением к скрипт-языку Tcl, для многих интерактивных приложений. Он будет установлен только во временный инструментарий.

- File

  Утилиты для определения типов файлов. Некоторым пакетам требуется, чтобы этот пакет был установлен.

- Findutils

  Программы для поиска файлов. Программы предоставляют способы для рекурсивного поиска файлов по дереву каталогов, создания, обслуживания и поиска в базе данных (как правильно поиск выполняется быстрее, чем рекурсивный поиск, но менее надёжен, если база данных не в актуальном состоянии).

- Flex

Утилиты для генерации программ, которые распознают шаблоны в тексте.Это GNU версия lex (lexical analyzer). Пакет необходим для сборки некоторых пакетов LFS.

- Gawk

  Программы для манипуляции с текстовыми файлами. Это GNU версия awk (Aho-Weinberg-Kernighan). Они используются в процедурах сборки для большинства пакетов.

- Gcc

  Набор компиляторов GNU, для таких языков, как C и C++. Поддержка других языков не предусмотрена в LFS

- GDBM

  Библиотека GNU Database Manager (менеджер баз данных GNU). Они используется пакетом Man-DB.

- Gettext

  Утилиты и библиотеки для работы с локализацией и интернационализацией необходимые для некоторых пакетов.

- Glibc

  Стандартная библиотека языка Си (GNU C Library). Linux программы не смогу без нее работать.

- GMP

  Библиотеки для математических вычислений и предоставляет полезные функции для вычислений с плавающей точкой. Он необходим для того, чтобы скомпилировать пакет Gcc.

- Gperf

  Программа, которая генерирует отличные хэш функции из набора ключей. Она необходима для пакета Eudev.

- Grep

  Программа которая принимает на вход строки, отвечающие заданному регулярному выражению, и выводит их, если вывод не отменён специальным ключом. Пакет используется в процедурах сборки для большинства пакетов.

- Groff

  Программы для обработки и форматирования текста. Одна из самых важных функций - форматирование man страниц.

- GRUB

  Загрузчик операционной системы (GRand Unified Bootloader). Этот пакет один из многих загрузчиков, но он является самым гибким.

- Gzip

  Программы для сжатия и распаковки файлов. Он необходим, чтобы выполнять распаковку многих пакетов LFS.

- Iana-etc

Данные для сетевых служб и протоколов. Он необходим для обеспечения правильных сетевых возможностей.

- Inetutils

  Программы programs для администрирования сетевых возможностей.

- Intltool

  Инструменты для извлечения переводимых строк из файлов с исходными кодами.

- IProute2

  Программы для работы с сетью по протоколам IPv4 и IPv6. Он был выбран из с других пакетов для работы с сетью из-за его поддержки IPv6 протокола.

- Kbd

  key-table файлы, утилиты для клавиатуры для не US наборов, а также консольные шрифты.

- Kmod

  Программы для администрирования модулей ядра Linux.

- Less

  Программа используемая для просмотра (но не изменения) содержимого текстовых файлов на экране. Она также используется пакетом Man-DB для просмотра страниц руководств.

- Libcap

  Интерфейсы пользовательского пространства для POSIX 1003.1e, доступные в ядрах Linux.

- Libelf

  Библиотека для обработки файлов формата ELF (Executable and Linkable Format — формат исполнимых и компонуемых файлов). формат двоичных файлов, используемый во многих современных UNIX-подобных операционных системах, таких как FreeBSD, Linux, Solaris и др. Также этот формат используется и во многих других системах. Большинство утилит доступны в других пакетах, но эта библиотека необходима для сборки ядра Linux используя конфигурацию по умолчанию (и наиболее эффективную).

- Libffi

  Переносимый, высокоуровневый интерфейс по различным соглашениям о вызовах. Программы во время компиляции могу не знать об аргументах, которые были переданы функции. Например, интерпретатору можно указать во время выполнения количество аргументов и указать их тип, для вызова функции. Libffi может использоваться в программах как "мост" от интерпретатора к скомпилированному коду.

- Libpipeline

  Библиотека для работы с подпроцессами гибким и удобным способом. Она необходима для пакета Man-DB.

- Libtool

  GNU libtool является общей библиотекой поддержки скриптов. Libtool скрывает сложность использования распределенных библиотек под последовательным, переносимым интерфейсом. Библиотека необходима для выполнения тестов других пакетов LFS.

- Linux Kernel

  Ядро операционной системы.

- M4

  Общий макропроцессор текста - полезный инструмент для выполнения сборки других программ.

- Make

  Программы которые автоматизируют процесс преобразования файлов из одной формы в другую. Чаще всего это компиляция исходного кода в объектные файлы и последующая компоновка в исполняемые файлы или библиотеки. Он необходим для сборки пакетов LFS.

- Man-DB

  Программы для поиска и просмотра страниц руководств. Он был выбран вместо пакета man благодаря превосходным возможностям интернационализации. Он содержит программу man.

- Man-pages

  Набор справочных руководств aLinux.

- Meson

  Инструменты для автоматизации сборки программ. Основная цель Meson - минимизировать затраты времени на конфигурирование системы сборки.

- MPC

  функции предназначенные для вычислений с плавающей запятой, целыми и рациональными числами с произвольной точностью. Он необходим для пакета GCC.

- MPFR

  Функции по работе с вычислениями с произвольной точностью. Они необходимы для пакета GCC.

- Ninja

  Утилита для сборки программ, фокусирующая на скорости. От других систем сборки она отличается двумя основными аспектами: для работы используется свой формат входных файлов, созданных системой сборки более высокого уровня, а также предназначена для быстрой сборки программ.

- Ncurses

  Библиотека, предназначенная для управления вводом-выводом на терминал, в том числе , библиотека позволяет задавать экранные координаты (в знакоместах) и цвет выводимых символов. Предоставляет программисту уровень абстракции, позволяющий не беспокоиться об аппаратных различиях терминалов и писать переносимый код. Она необходима для некоторых пакетов.

- Openssl

  Инструменты управления и библиотеки, относящиеся к криптографии. Они полезны для предоставления криптографических функций для других пакеты, а также для ядра Linux.

- Patch

  Программа предназначенная для переноса правок (изменений) между разными версиями текстовых файлов. Информация о правке обычно содержится в отдельном файле, называемом "заплаткой" (*patch*), "правкой" или "файлом правки" (англ. patch file). Подобный файл, как правило, создается с помощью другой утилиты Unix — diff, позволяющей автоматически извлечь информацию о различиях в тексте файлов. Он необходим для выполнения сборки некоторых пакетов LFS.

- Perl

  Высокоуровневый интерпретируемый динамический язык программирования общего назначения, он необходим для установки и выполнения тестов некоторых пакетов LFS.

- Pkg-config

  Утилита, предоставляющая интерфейс для получения информации об установленных программных библиотеках, включающую в себя параметры для C или C++ компилятора, параметры для компоновщика, а также версию пакета.

- Procps-NG

  Программы контроля за процессами. Этот набор программ может оказаться полезным системным администраторам. Он также используется скриптами загрузки LFS.

- Psmisc

  Программы для отображения информации о запущенных процессах.Этот набор программ может оказаться полезным системным администраторам.

- Python 3

  Высокоуровневый язык программирования общего назначения, ориентированный эффективности работы разработчика и читаемости кода

- Readline

  Библиотека интерфейса командной строки и обработки строк. Она используется командным интерпретатором Bash.

- Sed

  Sed - потоковый текстовый редактор (а также язык программирования), применяющий различные предопределённые текстовые преобразования к последовательному потоку текстовых данных. Он необходим для многих пакетов LFS, на этапе конфигурирования.

- Shadow

  Программы для работы с паролями безопасным способом.

- Sysklogd

  Пакет Sysklogd содержит программы для записи системных сообщений в журнал, например, сообщений ядра.

- Sysvinit

  Система инициализации init, которая является родителем всех остальных процессов в Linux системе.

- Tar

  Обеспечивает возможности архивирования и извлечения почти всех пакетов, используемых в LFS.

- Tcl

  "Командный язык инструментов" - скриптовый язык высокого уровня. Он необходим для выполнения тестов некоторых пакетов LFS, и будет установлен только во временный инструментарий.

- Texinfo

  Система документирования и язык разметки, позволяющие создавать документы в разных форматах из одного исходного текста. Она используется в процедурах установки многих пакетов LFS.

- Util-linux

  Стандартный набор служебных утилит командной строки, такие как - утилиты для работы с файловой системой, консолью, разделами, и сообщениями.

- Vim

  Текстовый редактор, созданный на основе более старого vi. Ныне это один из мощнейших текстовых редакторов с полной свободой настройки и автоматизации, возможными благодаря расширениям и надстройкам. Текстовый редактор является сугубо личным выбором для многих пользователей, и его можно заменить на любой другой, на ваш выбор.

- XML::Parser

  Модуль Perl который взаимодействует с Expat.

- XZ Utils

  Программы для сжатия и распаковки файлов. Она обеспечивает высокое сжатие и используется для распаковки пакетов форматов XZ и LZMA.

- Zlib

  Библиотека для сжатия и распаковки, которую используют некоторые программы.

Предпосылки

Сборка системы LFS - не простая задача. От вас потребуются знания в администрировании систем семейства Unix, для того, чтобы вы смогли устранять проблемы в процессе сборки, и правильно выполнять ввод требуемых команд. Как минимум, вы должны уметь пользоваться командной оболочкой, копировать и выполнять перемещение файлов и каталогов, просматривать списки каталогов и содержимое файлов и изменять текущий каталог. Также ожидается что у вас есть знания о процессе установки программного обеспечения в системах Linux.

Поскольку книга LFS предполагает, что у Вас уже есть необходимые навыки, различные источники справочной информации и поддержки LFS вряд ли смогут оказать вам помощь в этих вопросах. Вы обнаружите что Ваши вопросы по поводу базовых знаний, скорее всего останутся без ответа, или Вам будут указывать на ссылки по информации предварительного ознакомления.

Перед созданием системы LFS мы рекомендуем прочитать следующее:

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

  Это исчерпывающее руководство по сборке и установке “универсальных” Unix пакетов программ на системе Linux. Несмотря на то что книга написана достаточно давно, она дает качественную информацию по основным методам, необходимым для сборки и установки программного обеспечения.

- Руководство для новичков, как устанавливать программное обеспечение из исходных кодов *http://moi.vonos.net/linux/beginners-installing-from-source/*

  Руководство дает качественную информацию по основным методам, необходимым для сборки и установки программного обеспечения.

Оформление и типографские соглашения

Чтобы упростить работу, в книге используются некоторые соглашения по оформлению. Этот раздел содержит все необходимые примеры.

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

Этот текст необходимо набрать в командной строке в точности так, как показано. Если иное не сказано в тексте рядом. Это оформление также используется в объяснениях, когда указываются команды.

В некоторых случаях строка разделяется до двух или более линий с использованием символа обратного слэша в конце строки.

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

Обратите внимание, что после обратного слэша должнен быть перевод строки. Другие символы после - приведут к некорректному результату.

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

Вышеуказанная форма с текстом фиксированной ширины показывает вывод результатов на экран, а также показывает имена файлов, таких как `/etc/ld.so.conf` .

акцент

Эта форма текста используется в нескольких целях в книге. Его основная цель состоит в том, чтобы подчеркнуть важную информацию, на которую следует обратить особое внимание.

https://linuxfromscratch.ru/

Этот формат используется для ссылок на страницы проекта LFS, а также на внешние источники. Может включать справочную информацию, ссылки на загрузки и различные сайты.

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

Этот формат используется при создании файлов конфигурации. Первая команда создает файл`$LFS/etc/group` из всего содержимого текста, пока не встретит флаг конца файла (EOF). Обычно информация в таком блоке вводится так, как показано.

`<ЗАМЕНА ТЕКСТА>`

Этот формат используется для обозначения текста, который не должен вводиться так как отображен.

`[ДОПОЛНИТЕЛЬНЫЙ ТЕКСТ]`

Этот формат используется для обозначения текста, который можно внести опционально.

`passwd(5)`

Этот формат используется для ссылки на страницы руководств to refer to a specific manual (man) page. Число, которое указанно внутри скобок, указывает на конкретный раздел руководства. Например,**passwd** имеет две страницы. В книге LFS Per, эти две страницы будут расположены по адресу `/usr/share/man/man1/passwd.1` и `/usr/share/man/man5/passwd.5` . Когда в книгу указанно `passwd(5)` то необходимо ссылаться на файл `/usr/share/man/man5/passwd.5` . Команда **man passwd** отобразит первую страницу руководства по фразе “passwd”, которая будет ссылаться по пути `/usr/share/man/man1/passwd.1` . Например вам необходимо запустить команду **man 5 passwd** для прочтения указанной страницы. Следует отметить, что большинство страниц руководства не имеют дубликатов названий страниц в разных разделах. А значит, указание команды **man *<название программы>*** вполне достаточно.

Структура

Эта книга разделена на следующие части.

Часть I - Введение

Эта часть дает разъяснения о некоторых важных замечаниях о процессе установки LFS. Также, здесь будет представлена метаинформация о книге.

Часть II - Подготовка к сборке

Эта часть описывает как выполнить подготовку к процессу сборки—создать разделы на жестком диске, загрузить необходимые пакеты, и выполнить компиляцию временного инструментария.

Часть III - Сборка системы LFS

Эта часть описывает процесс сборки системы LFS. Процесс компиляции и установки каждого необходимого пакета, настройка загрузочных скриптов, и установку ядра. Результатом сборки будет система, которая является основой для дальнейшего расширения в Вашему желанию. В конце книги есть удобный перечень программ, библиотек и необходимых файлов, которые были установлены.

Опечатки и неточности

Программное обеспечение, используемое для создания системы LFS, постоянно обновляется и улучшается. Обновления безопасности и исправление ошибок программ могут появиться после выпуска очередной версии книги. Чтобы проверить, имеются ли версии пакетов

или инструкции в этом выпуске LFS и нуждаются ли они в каких либо модификациях для исправления уязвимостей системы безопасности или устранения других ошибок, пожалуйста, посетите *https://linuxfromscratch.ru/lfs/errata/9.0/* прежде чем приступить к сборке, Вы должны внести требуемые изменения и применить их к соответствующему разделу книги, до процесса создания системы LFS.

Part I. Введение

Chapter 1. Введение

1.1. Как выполнить сборку системы LFS

Сборка системы LFS будет выполнена с помощью уже установленного дистрибутива Linux (например Debian, OpenMandriva, Fedora, или openSUSE). Именно эта система (хост) будет служить отправной точкой. Необходимые компоненты хоста, такие как компилятор, компоновщик и командная оболочка будут использованы для сборки новой системы. Выберите режим или опцию “development (режим разработчика)” в процессе установки или конфигурирования хост системы чтобы использовать необходимые компоненты.

Иным вариантом, может быть установка отдельного дистрибутива на вашу машину, можно использовать LiveCD коммерческого дистрибутива.

Во второй главе описывается процесс создания новых разделов Linux. Это место, где будет выполняться компиляция, и куда будет установлена новая система LFS. В главе 3 описывается процесс настройки рабочего окружения и идут разъяснения о том, какие пакеты и патчи необходимо загрузить чтобы выполнить сборку LFS, а также каким образом их хранить на файловой системе. В четвертой главе обсуждается процесс настройки рабочего окружения. Внимательно изучите эту главу, так как в нем обсуждаются важные моменты, которые нужно знать до начала процесса сборки в пятой главе.

Пятая глава содержит информацию о пакетах, которые будут использоваться для создания базовой среды для разработки (набора инструментов), который, в свою очередь, будет использован для сборки окончательной системы LFS (Chapter 6). Некоторые из пакетов требуют разрешения циклической зависимости, например, чтобы выполнить сборку компилятора, вам нужен компилятор.

В пятой главе также объясняется каким образом выполнить первый проход сборки набора инструментов, используя пакеты Binutils и GCC (в первом проходе , обычно два этих пакета будут переустановленны). Следующим шагом будет инструкция по сборке пакета Glibc - библиотеки C. Библиотека C будет скомпилирована теми средствами, которые были установлены на этапе первого прохода сборки временного набора инструментов. Далее, следует второй проход сборки временного набора инструментов. На этот раз набор инструментов будет динамически скомпонован с собранной на предыдущем шаге библиотекой C. Оставшиеся пакеты будут использовать состояние временного набора инструментов после второго прохода. Когда все будет готово, процесс установки новой LFS системы больше не будет зависеть от хост-системы, кроме запущенного ядра Linux.

Изоляция процесса сборки новой системы от хост системы может выглядеть чрезмерной. Полное техническое обоснование, почему именно сделано так, представлено в главе.Section 5.2, “Технические примечания относительно временного набора инструментов”.

В главе Chapter 6, LFS система будет собрана. Команда **chroot** будет использована для смены корневого каталога на созданный ранее раздел LFS. После чего будет запущена новая командная оболочка. Эти действия напоминают процесс перезагрузки системы с указанием ядру что раздел LFS должен быть корневым. Система фактически не перезагружается, вместо чего используется команда **chroot**, потому что для создания загрузочной системы необходимо провести некоторые действия, которые на данном этапе еще не сделаны. Основным преимуществом является то, что изменение корневого раздела позволяет штатно использовать хост систему, пока будет происходить процесс сборки и конфигурирования LFS.

Когда базовая система LFS будет настроена, для окончания установки будут настроены Chapter 7. Ядро и системный загрузчик будут настроены в Chapter 8. Глава Chapter 9 содержит информацию о том, как расширить систему LFS и продолжить свой опыт в изучении, за пределами этой книги. После того, как все шаги, указанные в книги будут пройдены, можно будет перезагрузиться в новую LFS систему.

В этой главе очень кратко описан весь процесс работы. Подробно, всё будет рассмотрено в дальнейших главах. Многим вещам, которые могут казаться непонятными, будут даны уточнения и разъяснения, и по мере изучения этой книги всё станет на свои места.

1.2. Что нового в релизе

Ниже приведен список обновлений пакетов, выпущенных с предыдущего выпуска книги.

Обновлено:

-
- Bc 2.1.3
- Bison-3.4.1
- Bzip2-1.0.8
- E2fsprogs-1.45.3
- Eudev-3.2.8
- Expat-2.2.7
- File-5.37
- Gawk-5.0.1
- GCC-9.2.0
- Gettext-0.20.1
- Glibc-2.30
- GRUB-2.04
- IPRoute2-5.2.0
- Kbd-2.2.0
- Less-551
- Libcap-2.27
- Libelf-0.177 (from elfutils)
- Linux-5.2.8
- Man-DB-2.8.6.1
- Openssl-1.1.1c
- Perl-5.30.0

- Python-3.7.4
- Shadow-4.7
- SysVinit-2.95
- Tar-1.32
- Texinfo-6.6
- Tzdata-2019b
- Util-Linux-2.34
- Vim-8.1.1846

Добавлено:

•

Удалено:

•

1.3. Журнал изменений

Эта версия - 9.0 книги Linux From Scratch, от 1 Сентября, 2019. Если этой книге более чем пол года, скорее всего, более свежая версия уже доступна. Чтобы найти новую версию, можно воспользоваться зеркалами проекта *https://linuxfromscratch.ru/mirrors.html* .

Ниже представлен перечень изменений, сделанных с предыдущей версии.

Changelog Entries:

- 2019-09-01
  - [bdubbs] - LFS-9.0 released.
- 2019-08-14
  - [bdubbs] - Update to vim-8.1.1846. Fixes *#4500*.
  - [bdubbs] - Update to elfutils-0.177. Fixes *#4516*.
  - [bdubbs] - Update to gcc-9.2.0. Fixes *#4514*.
  - [bdubbs] - Update to bc-2.1.3. Fixes *#4513*.
  - [bdubbs] - Update to man-db-2.8.6.1. Fixes *#4512*.
  - [bdubbs] - Update to linux-5.2.8. Fixes *#4511*.
- 2019-08-04
  - [bdubbs] - Fix a problem introduced by linux-5.2 by adding an include file to a glibc header.
- 2019-08-03
  - [bdubbs] - Update to linux-5.2.5. Fixes *#4505*.
  - [bdubbs] - Update to kbd-2.2.0. Fixes *#4507*.
  - [bdubbs] - Update to glibc-2.30. Fixes *#4508*.

- [bdubbs] - Update to man-pages-5.02. Fixes *#4509*.
- 2019-07-21
  - [bdubbs] - Update to linux-5.2.2. Fixes *#4504*.
  - [bdubbs] - Update to bc-2.1.1. Fixes *#4503*.
  - [bdubbs] - Update to kbd-2.1.0. Fixes *#4502*.
  - [bdubbs] - Update to e2fsprogs-1.45.3. Fixes *#4501*.
- 2019-07-14
  - [bdubbs] - Fix testing of binutils-2.32 gold linker. Fixes *#4498*.
  - [bdubbs] - Update to tzdata-2019b. Fixes *#4492*.
  - [bdubbs] - Update to python3-3.7.4. Fixes *#4496*.
  - [bdubbs] - Update to meson-0.51.1. Fixes *#4497*.
  - [bdubbs] - Update to iproute2-5.2.0. Fixes *#4495*.
  - [bdubbs] - Update to grub-2.04. Fixes *#4494*.
  - [bdubbs] - Update to linux-5.2.1. Fixes *#4493*.
  - [bdubbs] - Update to bc-2.1.0. Fixes *#4436*.
  - [bdubbs] - Update to bzip2-1.0.8. Fixes *#4499*.
- 2019-06-29
  - [bdubbs] - Properly initialize a data structure in OpenSSL to avoid valgrind uninitialized value errors. Fixes *#4491*.
  - [bdubbs] - Update to meson-0.51.0. Fixes *#4483*.
  - [bdubbs] - Update to gawk-5.0.1. Fixes *#4486*.
  - [bdubbs] - Update to expat-2.2.7. Fixes *#4488*.
  - [bdubbs] - Update to linux-5.1.15. Fixes *#4487*.
  - [bdubbs] - Update to sysvinit-2.95. Fixes *#4484*.
  - [bdubbs] - Update to bzip2-1.0.7. Fixes *#4490*.
- 2019-06-24
  - [renodr] - Fixed issue with installing Check's documentation in a versioned directory. Thanks goes to Ryan Marsaw for the report. This was fixed by removing the unrecognized/unused --docdir and replacing it with a "docdir=" in the make install command.
- 2019-06-18
  - [renodr] - Update to linux-5.1.11. Fixes the SOCK PANIC issue. Fixes *#4485*.
- 2019-06-16
  - [bdubbs] - Update to vim-8.1.1535. Fixes *#4482*.
  - [bdubbs] - Update to shadow-4.7. Fixes *#4481*.
  - [bdubbs] - Update to linux-5.1.10. Fixes *#4478*.
  - [bdubbs] - Update to less-551. Fixes *#4477*.

- [bdubbs] - Update to util-linux-2.34. Fixes *#4462*.
- [bdubbs] - Remove eudev instructions referring to /tools. Fixes *#4480*.

- 2019-06-08
  - Make it so that the instructions for removing symlinks before building Util-Linux in Chapter 6 are only visible in systemd.
- 2019-05-03
  - [bdubbs] - Update to gcc-9.1.0. Fixes *#4463*.
  - [bdubbs] - Update to linux-5.0.11. Fixes *#4461*.
- 2019-04-23
  - [bdubbs] - Apply a change to allow Perl to build correctly when building with the latest versions of gcc.
- 2019-04-20
  - [bdubbs] - Update to perl-5.28.2. Fixes *#4460*.
  - [bdubbs] - Update to meson-0.50.1. Fixes *#4459*.
  - [bdubbs] - Update to linux-5.0.9. Fixes *#4458*.
  - [bdubbs] - Update to libcap-2.27. Fixes *#4457*.
- 2019-04-15
  - [bdubbs] - Update bzip2 url. Fixes *#4450*.
  - [bdubbs] - Update to util-linux-2.33.2. Fixes *#4454*.
  - [bdubbs] - Update to linux-5.0.7. Fixes *#4449*.
  - [bdubbs] - Update to gawk-5.0.0. Fixes *#4455*.
- 2019-03-27
  - [bdubbs] - Revert to meson-0.49.2.
- 2019-03-26
  - [bdubbs] - Update to tzdata2019a. Fixes *#4448*.
  - [bdubbs] - Update to Python-3.7.3. Fixes *#4447*.
- 2019-03-25
  - [bdubbs] - Update to iproute2-5.0.0. Fixes *#4446*.
  - [bdubbs] - Update to linux-5.0.4. Fixes *#4444*.
  - [xry111] - Use -ffile-prefix-map instead of -isystem and symlinks in the Glibc build to simplify the instruction.
- 2019-03-13
  - [xry111] - Update contents and short descriptions of packages. Fixes *#4443*.
- 2019-03-12
  - [bdubbs] - Update to meson-0.50.0. Fixes *#4442*.
  - [bdubbs] - Update to coreutils-8.31. Fixes *#4441*.

    - [bdubbs] - Update to linux-5.0.1. Fixes *#4440*.
    - [bdubbs] - Update to man-pages-5.00. Fixes *#4439*.
    - [bdubbs] - Update to e2fsprogs-1.45.0. Fixes *#4438*.
- 2019-03-05
    - [bdubbs] - Update to linux-5.0. Fixes *#4437*.
- 2019-03-01
    - [bdubbs] - Update to texinfo-6.6. Fixes *#4427*.
    - [bdubbs] - Update to tar-1.32. Fixes *#4431*.
    - [bdubbs] - Update to sysvinit-2.94. Fixes *#4433*.
    - [bdubbs] - Update to openssl-1.1.1b. Fixes *#4435*.
    - [bdubbs] - Update to gcc-8.3.0. Fixes *#4430*.
    - [bdubbs] - Update to linux-4.20.13. Fixes *#4434*.
- 2019-03-01
    - [bdubbs] - LFS-8.4 released.

1.4. Ресурсы

1.4.1. Часто задаваемые вопросы

Если в процессе сборки LFS вы столкнулись с ошибками, у вас появились вопросы, или считаете, что в книге есть опечатка, пожалуйста, начните с раздела часто задаваемых вопросов (FAQ), который находится по адресу *https://linuxfromscratch.ru/faq/*.

1.4.2. Списки рассылки

На сервере `linuxfromscratch.org` есть несколько списков рассылки, которые используются для разработки проекта LFS. Эти списки включают в себя списки разработки и поддержки. Если FAQ не помог в решении проблем, следующим шагом будет изучение списков рассылок по адресу *https://linuxfromscratch.ru/search.html*.

Информация о содержании списков рассылки, как подписаться, архивы и дополнительная информация находится по адресу *https://linuxfromscratch.ru/mail.html*.

1.4.3. IRC

Некоторые члены сообщества предлагают помощь через протокол прикладного уровня для обмена сообщениями в режиме реального времени (IRC). Перед тем, как его использовать, убедитесь что на ваш вопрос не нашлось ответов в FAQ или архиве списков рассылок. По адресу `irc.freenode.net` канал по поддержке называется #LFS-support.

1.4.4. Зеркала проекта

Проект LFS имеет зеркала. Пожалуйста, перейдите по ссылке *https://linuxfromscratch.ru/mirrors.html* , чтобы ознакомиться с полным перечнем зеркал.

1.4.5. Контактная информация

Направляйте ваши вопросы и комментарии в необходимый список рассылки. (смотреть выше)

1.5. Помощь

Если у Вас возникла проблема или вопрос, в процессе чтения книги, посетите раздел часто задаваемых вопросов, который расположен по адресу *https://linuxfromscratch.ru/faq/#generalfaq*. Как правило здесь можно найти ответы на большинство вопросов, однако, если ответов не нашлось, по ссылке *https://linuxfromscratch.ru/hints/downloads/files/errors.txt* можно найти некоторые рекомендации по устранению проблем.

Если вам не удалось найти решение проблемы в разделе часто задаваемых вопросов, поищите информацию по спискам рассылок по адресу *https://linuxfromscratch.ru/search.html*.

У нас также есть замечательное сообщество LFS, которое готово предложить помощь через списки рассылки и IRC (ознакомьтесь с разделом Section 1.4, “Ресурсы”). Однако, мы получаем много вопросов каждый день, и на многие из них легко можно найти ответ в разделе часто задаваемых вопросов, или в списках рассылки. Для того, чтобы мы смогли помочь вам лучше, необходимо для начала исследовать проблему самостоятельно. Это позволит сосредоточится на более необычных проблемах. Если поиск ответа не дает результатов, вам необходимо включить всю необходимую информацию , которая указана ниже, в ваш запрос о помощи.

1.5.1. Что нужно сделать

Кроме краткого и содержательного описания проблемы, к вашему вопросу необходимо добавить следующее:

- Версию этой книги (в данном случае 9.0)
- Информацию о дистрибутиве и его версия, на котором выполнялось создание системы LFS
- Вывод сценария Требования к хост-системе
- Пакет или раздел где возникла проблема.
- Точное сообщение о возникшей ошибке, или её признаки.
- Отметку о сделанных отклонениях от инструкций книги

Note

Отклонение от инструкций в книге еще не означает то что мы не окажем вам помощь. В конце концов - процесс создания системы LFS может быть основан на ваших предпочтениях, и иметь информацию об изменениях, которые были внесены, поможет определить причины проблемы.

1.5.2. Проблемы со скриптами конфигурирования (Configure)

Если что-нибудь пошло не так, в процессе работы скрипта **configure**, следует изучить файл `config.log` . Этот файл может содержать информацию о тех ошибках, которые не отображаются на экране в процессе выполнения скрипта. Вам необходимо включить строки, содержащие информацию об ошибке когда будете задавать вопросы.

1.5.3. Проблемы компиляции

Вывод на экран и содержимое различных файлов могут быть полезными для определения причин проблем компиляции. Также, могут быть полезным вывод информации выполнения скрипта **configure** и **make**. Не нужно включать весь вывод, выберите только информативный фрагмент. Ниже приведен пример какая информация может быть включена с результата вывода скрипта **make**:

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

В этом случае, многие включают следующую информацию:

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

Этой информации недостаточно чтобы определить проблему, потому что эта строка говорит о том, что произошла ошибка и ни о чем более. Поэтому, необходимо сохранить всю информацию, включая вводимые команды и текст ошибки или сообщения.

Отличная статья о том, как правильно составить вопрос доступна по ссылке *http://catb.org/~esr/faqs/smart-questions.html*. Прочтите и следуйте подсказкам которые содержаться в этом документе, чтобы шанс получения помощи был выше.

Part II. Подготовка к сборке

Chapter 2. Подготовка хост-системы

2.1. Введение

В этой главе, инструменты хост-системы необходимые для сборки LFS, будут проверены и при необходимости установлены. Будет выполнена подготовка дискового раздела где будет размещаться система LFS, создание файловой системы и её монтирование.

2.2. Требования к хост-системе

Ваша хост система должна иметь следующее программное обеспечение с минимальными версиями, которые будут указаны. На современных дистрибутивах проблем не должно быть, однако, необходимо знать, что некоторые дистрибутивы хранят заголовочные файлы в отдельных пакетах, как правило, в таком формате: “<package-name>-devel” или “<package-name>-dev”. Убедитесь что они были установлены, в случае если ваш дистрибутив их предоставляет.

Более ранние версии из списка программного обеспечения могут работать, но этот момент не проверялся (тестирование не производилось).

- **Bash-3.2** (/bin/sh должен быть символической или жесткой ссылкой на bash)
- **Binutils-2.25** (Версия старше чем 2.32 не рекомендуется, потому что тестирование таковой не проводилось.
- **Bison-2.7** (/usr/bin/yacc должен быть ссылкой на bison или небольшим скриптом который выполняет bison)
- **Bzip2-1.0.4**
- **Coreutils-6.9**
- **Diffutils-2.8.1**
- **Findutils-4.2.31**
- **Gawk-4.0.1** (/usr/bin/awk должен быть ссылкой на gawk)
- **GCC-6.2** включая компилятор Си++ , **g++** (Версия старше чем 9.2.0 не рекомендуется, потому что тестирование таковой не проводилось.)
- **Glibc-2.11** (Версия старше чем 2.30 не рекомендуется, потому что тестирование таковой не проводилось.)
- **Grep-2.5.1a**
- **Gzip-1.3.12**
- **Ядро Linux 3.2**

  Причина минимальной версии ядра - указание такой версии при сборке пакета glibc в главе 6, которая является рекомендацией разработчиков. Она также необходима для пакета udev.

  Если версия ядра хост системы более ранняя, чем 3.2 вам придется заменить его на более свежую. Существует как минимум два способа как это сделать. Сперва посмотрите, предоставляет ли дистрибутив хост системы ядро версии 3.2 или более позднюю. Если этот так - нужно его обновить. В ином случае, вы можете выполнить сборку ядра самостоятельно. Инструкции для компиляции ядра и настройка загрузчика представлены в главе Chapter 8 (При условии что в хост системе используется загрузчик GRUB).

- **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-4.7**
- **Xz-5.0.0**

Important

Обратите внимание, что упомянутые выше символические ссылки необходимы для создания системы LFS используя инструкции, содержащиеся в этой книге. Символические ссылки которые указывают на программы (такие как dash, mawk, и т.д.) могут работать, но они не тестировались и не поддерживаются командой разработчиков LFS и возможно потребуются отклонения от инструкций или применение дополнительных патчей для некоторых пакетов.

Чтобы узнать, имеет ли ваша хост-система все соответствующие версии и возможность компиляции программ, выполните следующие действия:

```
  cat > version-check.sh << "EOF"
   #!/bin/bash
# Simple script to list version numbers of critical development tools
export LC_ALL=C
bash --version | head -n1 | cut -d" " -f2-4 MYSH=$(readlink -f /bin/sh)
echo "/bin/sh -> $MYSH"
echo $MYSH | grep -q bash || echo "ERROR: /bin/sh does not point to bash"
unset MYSH

echo -n "Binutils: "; ld --version | head -n1 | cut -d" " -f3-
bison --version | head -n1

if [ -h /usr/bin/yacc ]; then
  echo "/usr/bin/yacc -> `readlink -f /usr/bin/yacc`";
elif [ -x /usr/bin/yacc ]; then
  echo yacc is `/usr/bin/yacc --version | head -n1`
else
  echo "yacc not found"
fi

bzip2 --version 2>&1 < /dev/null | head -n1 | cut -d" " -f1,6-
echo -n "Coreutils: "; chown --version | head -n1 | cut -d")" -f2
diff --version | head -n1
find --version | head -n1
gawk --version | head -n1

if [ -h /usr/bin/awk ]; then
  echo "/usr/bin/awk -> `readlink -f /usr/bin/awk`";
elif [ -x /usr/bin/awk ]; then
  echo awk is `/usr/bin/awk --version | head -n1`
else
  echo "awk not found"
fi
```

```
gcc --version | head -n1
g++ --version | head -n1
ldd --version | head -n1 | cut -d" " -f2-  # glibc version
grep --version | head -n1
gzip --version | head -n1
cat /proc/version
m4 --version | head -n1
make --version | head -n1
patch --version | head -n1
echo Perl `perl -V:version`
python3 --version
sed --version | head -n1
tar --version | head -n1
makeinfo --version | head -n1  # texinfo version
xz --version | head -n1
```

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

bash version-check.sh
```

2.3. Этапы сборки системы LFS

Процедуру сборки системы LFS необходимо выполнить за один сеанс, и предполагается, что система не будет выключена на протяжении всего процесса. Но это не означает, что сборка должна быть выполнена за один приём. Проблема в том, что некоторые процедуры должны быть выполнены повторно после перезагрузки системы.

2.3.1. Главы 1–4

Эти главы должны быть выполнены на хост-системе. После перезагрузки, обратите особое внимание на следующее:

- Процедуры выполняемые из-под корневого пользователя (root) в главе 2.4 требуют, чтобы была правильно установлена переменная окружения LFS *ДЛЯ КОРНЕВОГО ПОЛЬЗОВАТЕЛЯ (ROOT)*.

2.3.2. Глава 5

- Раздел /mnt/lfs должен быть смонтирован.
- *ВСЕ* инструкции в Главе 5 ы должны выполняться из-под пользователя *lfs*. Команда **su - lfs** должны быть выполнена перед выполнением любой задачи в главе 5.
- Выполнение процедур в Section 5.3, “Общие инструкции по компиляции” очень важно. Если есть сомнения по установке пакета, убедитесь что каталог, куда был ранее распакован архив - удален. Далее извлеките архив пакета и выполните инструкции в разделе.

2.3.3. Главы 6–8

- Раздел /mnt/lfs должен быть смонтирован.
- Когда будет выполняться команда chroot, переменная окружения LFS должна быть указана корневому пользователю (root). В других случаях она не требуется.
- Виртуальные файловые системы должны быть смонтированы. Это можно сделать до или после ввода chroot путем изменения в виртуальном терминале хоста и, как root, запустить команды Section 6.2.2, “Монтирование и заполнение каталога /dev” и Section 6.2.3, “Монтирование виртуальных файловых систем ядра”.

2.4. Создание нового раздела

Как большинство других операционных систем, LFS обычно устанавливается на выделенный раздел. Рекомендуемый подход к созданию системы LFS, заключается в использовании доступного пустого раздела или, если у вас есть неразмеченное пространство, вы можете использовать его.

Минимальная система требует раздела емкостью 6 гигабайт. Этого размера будет достаточно для хранения всех архивов с исходными кодами всех пакетов и их последующей компиляции. Однако, если в дальнейшем вы захотите чтобы система LFS была первичной Linux системой, и предполагается, что будут установлены другие пакеты, которые потребуют дополнительного дискового пространства, создание раздела ёмкостью в 20 гигабайт будет более разумным. Сама система LFS не займет столько места. Большая часть пространства будет использована для временного хранения и для её последующего расширения. Кроме того, при компиляции пакетов потребуется много дополнительного дискового пространства, которое будет освобождено после установки пакета.

Так как доступной оперативной памяти (RAM) может быть не достаточно, рекомендуется использовать небольшой дисковый раздел swap. Он будет использоваться ядром для хранения редко используемых данных и освобождать оперативную память для активных процессов. Раздел swap для системы LFS можно использовать тот, который используется хост-системой, и создавать новый раздел - необязательно.

Start a disk partitioning program such as **cfdisk** or **fdisk** with a command line option naming the hard disk on which the new partition will be created—for example `/dev/sda` for the primary disk drive. Create a Linux native partition and a swap partition, if needed. Please refer to `cfdisk(8)` or `fdisk(8)` if you do not yet know how to use the programs.

Note

Для опытных пользователей возможны другие варианты разбивки диска. Система LFS может быть установлена на *RAID* массив, или на *LVM* том. Однако, некоторые из этих опций требуют пакет *initramfs*, которые обсуждается за пределами этой книги, и эти опции разбивки не рекомендуются для первого изучения этой книги.

Запомните наименования созданных разделов (например sda5). Информация в книге будет ссылаться на этот дисковый раздел. Также помните о назначении swap раздела. Эти названия понадобятся позднее для указания их в файле `/etc/fstab` .

2.4.1. Другие вопросы по созданию разделов

Вопросы и рекомендации по созданию разделов часто публикуются в списках рассылок LFS. Это очень субъективная тема. По умолчанию для большинства дистрибутивов создание раздела заключается в использовании всего диска, за исключением одного небольшого раздела подкачки. Это не является оптимальным для LFS по нескольким причинам. Это уменьшает гибкость, делает совместное использование данных в нескольких дистрибутивах или сборках LFS сложнее, усложняет процесс резервного копирования, и требует больше времени и может тратить дисковое пространство менее эффективно.

2.4.1.1. Корневой раздел

Корневой раздел LFS (не путать с каталогом `/root`) размером в 10 гигабайт является хорошим компромиссом для многих систем. Такого размера будет достаточно для сборки LFS и большинства пакетов BLFS, но при этом, он достаточно мал. Для проведения экспериментов можно использовать несколько разделов.

2.4.1.2. Раздел подкачки

Многие дистрибутивы автоматически создают раздел подкачки. Как правило, рекомендуемый размер - в два раза больше, чем объем оперативной памяти, однако такая необходимость бывает редко. Если дисковое пространство ограничено, установите размер в два гигабайта и контролируйте размер занимаемого места файла подкачки.

Использование файла подкачки - не очень хорошо. Легко сразу понять что система использует файл подкачки - активность обращения к жесткому диску, где расположен файл подкачки будет высока и реакция на выполнение команд тоже будет другой. Если использование файла подкачки приводит к ненормальному падению производительности, лучшим решением будет увеличить размер оперативной памяти вашей компьютера.

2.4.1.3. Радел GRUB BIOS

Если *загрузочный диск* был размечен с помощью GUID Partition Table (GPT), в таком случае маленький раздел, обычно около одного мегабайта, должен быть создан, если он еще не существует. Этот раздел не будет форматирован и должен быть доступен для GRUB, для использования в процессе установки загрузчика. Метка этого раздела обычно называется 'BIOS Boot' если используется утилита **fdisk** или иметь код *EF02* если используется утилита **gdisk**.

Note

Раздел Grub Bios должен быть на диске, который BIOS будет использовать для загрузки системы. И не обязательно, что раздел, на котором будет находится корневой каталог LFS, будет на том же диске. Диски в системе могут использовать различные таблицы разделов. Требование для этого раздела зависит только от типа таблицы - загрузочного раздела.

2.4.1.4. Convenience Partitions

Существует несколько других разделов, которые не требуются, но должны учитываться при проектировании и разбивке разделов. Приведенный ниже перечень не является всеобъемлющим, и представлен для краткого ознакомления.

- /boot – Настоятельно рекомендуется. Используйте этот раздел для хранения Linux ядер и загрузочной информации. Чтобы минимизировать потенциальные проблемы загрузки с большими дисками, сделайте первый физический раздел размером в 100 мегабайт, которого будет достаточно.
- /home – Настоятельно рекомендуется. Раздел для домашнего каталога пользователей для нескольких дистрибутивов или сборок LFS. Размер как правило необходимо указать достаточно большой, в зависимости от доступного места на диске.

- /usr –Отдельный раздел /usr обычно используется для тонких клиентов и бездисковых рабочих станций. Использование этого каталога как отдельного раздела - обычно не требуется для LFS. Размер около пяти гигабайт подойдёт для большинства установок.
- /opt – Этот каталог будет полезен в основном для BLFS, где множественная установка больших пакетов, таких как Gnome или KDE может быть выполнена не в иерархию каталогов /usr. Размера от пяти до десяти гигабайт будет достаточно.
- /tmp – Отдельный каталог / tmp встречается редко,и полезен для работы тонких клиентов. Если используется этот раздел, он может не превышать пары гигабайт.
- /usr/src – Этот раздел полезно иметь для сохранения исходных файлов BLFS и выполнять обмен между сборками LFS. Он также может использоваться как место для сборки пакетов BLFS. 30 - 50 гигабайт будет достаточно.

Любой раздел, который должен быть смонтирован во при загрузке нужно указать в файле `/etc/fstab` . Информация о том, как указывать разделы в этой файле описано в главе Section 8.2, “Создание файла /etc/fstab”.

2.5. Создание файловой системы на разделе

На данный момент, были созданы пустые разделы. Файловые системы могут быть созданы. LFS может использовать любые файловые системы, которые могут быть распознаны ядром Linux. В основном это файловые системы ext3 и ext4. Выбор файловой системы может зависеть от характеристик хранимых файлов и размера раздела. Например:

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

ext3
: Это обновленная файловая система ext2, включающая в себя журнал, для восстановления в случае некорректного выключения. Обычно используется как файловая система общего назначения.

ext4
: Является на данный момент последней версией файловой системы семейств ext. Она поддерживает много новых возможностей таких как nano-second метки времени, создание и использование очень больших файлов (16 Терабайт), и улучшения быстродействия.

Другие файловые системы, такие как FAT32, NTFS, ReiserFS, JFS, и XFS полезны для конкретных задач. Подробную информацию по о файловых системах можно изучить по ссылке *https://ru.wikipedia.org/wiki/%D0%A1%D1%80%D0%B0%D0%B2%D0%BD%D0%B5%D0%BD%D0%B8%D0%B5_%D1%84%D0%B0%D0%B9%D0%BB%D0%BE%D0%B2%D1%8B%D1%85_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC*.

LFS предполагает, что корневая файловая система будет ext4. Чтобы выполнить форматирование раздела используя эту файловую систему, выполните следующую команду:

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

Если вы используете существующий swap раздел, форматировать его не требуется. Если был создан новый раздел swap его необходимо инициализировать командой:

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

Replace *<yyy>* with the name of the swap partition.

2.6. Настройка переменной окружения $LFS

Во многих главах и разделах книги присутствует переменная окружения LFS. До выполнения процесса сборки, необходимо убедиться в том, что переменная определена. Переменная должна хранить путь до каталога, где будет выполняться процесс сборки LFS системы. Например, мы будем использовать каталог - `/mnt/lfs` . Несомненно, вы можете изменить путь на ваше усмотрение. Если сборка выполняется на отдельном разделе, каталог может быть точкой монтирования раздела. Определите каталог, где будет храниться система LFS и выполните следующую команду:

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

Имея такую переменную, будет очень удобно выполнять команды, например **mkdir -v $LFS/tools** можно набирать буквально. Командная оболочка автоматически заменит символы “$LFS” в “/mnt/lfs” (или иным значением, которые вы присвоили этой переменной) во время обработки команды.

Caution

Не забудьте проверить что переменная LFS устанавливается каждый раз, когда выходите и заново заходите в рабочую оболочку (например, при выполнении команды **su** в `root` или другого пользователя). Проверьте что переменная установлена правильно. Введите команду чтобы в этом убедиться:

```
echo $LFS
```

Убедитесь что вывод отображает путь к тому каталогу, где будет выполнена сборка системы LFS (`/mnt/lfs`). Если вывод неправильный, используйте команду которая обсуждалась выше, для установки переменной LFS требуемого значения пути к системе LFS.

Note

Ещё один способ удостовериться что значение переменной корректное - указать ее значение в файле `.bash_profile` в домашнем каталоге и в каталоге `/root/.bash_profile` . Кроме того, оболочка, которая указанна в файле `/etc/passwd` для всех пользователей, которым необходимо наличие переменной LFS нужна bash для гарантии того, что файл `/root/.bash_profile` используется процессом авторизации в систему.

Есть ещё один метод который используется при входе в хост-систему. Если при входе в хост-систему используется графический менеджер, пользовательский файл `.bash_profile` не будет корректно использоваться при запуске виртуального терминала. В этом случае, необходимо добавить команду export в файл `.bashrc` для необходимого пользователя а также для пользователя root. К тому же некоторые дистрибутивы имеют инструкции не запускать файл `.bashrc` в не интерактивном вызове bash. Обязательно добавьте export перед тем, как проверить работу в не интерактивном режиме.

2.7. Монтирование нового раздела

Теперь, когда файловые системы были созданы, необходимо сделать разделы доступными. Чтобы это сделать, необходимо примонтировать разделы в выбранные точки монтирования. В книге предполагается что файловая система монтируется в каталог, который был указан в переменной окружения LFS, как было описано в предыдущем разделе книги.

Создадим точку монтирования и выполним монтирование файловой системы LFS, выполнив команду:

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

Замените *<xxx>* наименованием форматированного раздела LFS.

Если вы используете несколько разделов (например один для `/`, другой для `/usr`), выполните их монтирование выполнив команды:

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

Замените *<xxx>* and *<yyy>* соответствующим наименованием раздела.

Убедитесь что монтирование не было выполнено с ограниченными правами (такими как `nosuid` or `nodev` опциями). Выполните команду **mount** без каких либо параметров чтобы посмотреть какие опции заданы для смонтированных разделов LFS. Если опции `nosuid` и (или) `nodev` установлены, разделы необходимо размонтировать и выполнить монтирование заново.

Warning

В приведенных выше инструкциях предполагается, что вы не будете перезапускать ваш компьютер в процессе сборки LFS. Если вы выключите компьютер, вам потребуется выполнять монтирование повторно каждый раз, после перезагрузки хост-системы. Вы можете отредактировать файл /etc/fstab, чтобы разделы автоматически монтировались после перезагрузки. Например:

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

Если вы используете другие разделы, добавьте их также.

Если вы используете swap раздел, убедитесь что он включен, используя команду **swapon**:

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

Замените *<zzz>* названием раздела swap.

Теперь, когда место для работы организовано, самое время приступить к загрузке пакетов.

Chapter 3. Пакеты и патчи

3.1. Введение

В этой главе будут инструкции о том какие пакеты и патчи необходимо загрузить для создания системы LFS. Указанные версии пакетов проверены и работают в текущей версии этой книги. Мы не рекомендуем использовать новые версии пакетов, потому что инструкции по сборке пакета одной версии, может не работать с более новой. Также могут возникнуть проблемы, для решения которых потребуются обходные пути. Такие проблемы решаются в нестабильных релизах книги.

Источники загрузки пакетов могут быть недоступны. Если источник изменился со времени публикации этой версии книги, Google (*https://www.google.com/*) является удобным инструментом для поиска пакетов. Если поиск не увенчался успехом, попробуйте один из альтернативных способов загрузки, обсуждаемых в *https://linuxfromscratch.ru/lfs/packages.html#packages*.

Загруженные пакеты и патчи необходимо где-нибудь сохранить. Нужен рабочий каталог, в котором можно будет распаковывать пакеты и выполнять их настройку и компиляцию. Каталог `$LFS/sources` может быть использован как место для хранения, а также как место для настройки и компиляции. Используя этот каталог, необходимые элементы будут расположены и доступны на всех этапах создания системы LFS.

Чтобы создать такой каталог, выполните следующую команду как пользователь `root`, до начала процесса загрузки пакетов и патчей:

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

Сделайте этот каталог доступным для записи и липким. “Липким” означает, что даже если у некоторых пользователей есть разрешение на запись в этот каталог, то только владелец сможет удалить содержимое каталога. следующая команда позволит это сделать:

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

Самый простой способ загрузки всех требуемых пакетов и патчей - воспользоваться файлом *wget-list*. Далее его можно передать как параметр программе **wget**. Например:

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

Начиная с 7 версии, есть два отдельных файла - *md5sums*, которые могут понадобится для проверки пакетов. Поместите этот файл в каталог `$LFS/sources` и выполните команду:

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

3.2. Все пакеты

Загрузите или иным образом получите следующие пакеты с указанными версиями:

• Acl (2.2.53) - 513 KB:
Ссылка на загрузку: *https://download.savannah.gnu.org//releases/acl/acl-2.2.53.tar.gz*
Контрольная сумма MD5: `007aabf1dbb550bcddde52a244cd1070`

• Attr (2.4.48) - 457 KB:
Домашняя страница: *https://savannah.nongnu.org/projects/attr*
Ссылка на загрузку: *https://download.savannah.gnu.org//releases/attr/attr-2.4.48.tar.gz*
Контрольная сумма MD5: bc1e5cb5c96d99b24886f1f527d3bb3d

• Autoconf (2.69) - 1,186 KB:
Домашняя страница: *https://www.gnu.org/software/autoconf/*
Ссылка на загрузку: *https://ftp.gnu.org/gnu/autoconf/autoconf-2.69.tar.xz*
Контрольная сумма MD5: 50f97f4159805e374639a73e2636f22e

• Automake (1.16.1) - 1,499 KB:
Домашняя страница: *https://www.gnu.org/software/automake/*
Ссылка на загрузку: *https://ftp.gnu.org/gnu/automake/automake-1.16.1.tar.xz*
Контрольная сумма MD5: 53f38e7591fa57c3d2cee682be668e5b

• Bash (5.0) - 9,898 KB:
Домашняя страница: *https://www.gnu.org/software/bash/*
Ссылка на загрузку: *https://ftp.gnu.org/gnu/bash/bash-5.0.tar.gz*
Контрольная сумма MD5: 2b44b47b905be16f45709648f671820b

• Bc (2.1.3) - 233 KB:
Домашняя страница: *https://github.com/gavinhoward/bc*
Ссылка на загрузку: *https://github.com/gavinhoward/bc/archive/2.1.3/bc-2.1.3.tar.gz*
Контрольная сумма MD5: 2a882dc39c0fb8e36c12f590d54cc039

• Binutils (2.32) - 20,288 KB:
Домашняя страница: *https://www.gnu.org/software/binutils/*
Ссылка на загрузку: *https://ftp.gnu.org/gnu/binutils/binutils-2.32.tar.xz*
Контрольная сумма MD5: 0d174cdaf85721c5723bf52355be41e6

• Bison (3.4.1) - 2,147 KB:
Домашняя страница: *https://www.gnu.org/software/bison/*
Ссылка на загрузку: *https://ftp.gnu.org/gnu/bison/bison-3.4.1.tar.xz*
Контрольная сумма MD5: 201286a573b12da109df96282fe4ff4a

• 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 (0.12.0) - 747 KB:
Домашняя страница: *https://libcheck.github.io/check*
Ссылка на загрузку: *https://github.com/libcheck/check/releases/download/0.12.0/check-0.12.0.tar.gz*
Контрольная сумма MD5: 31b17c6075820a434119592941186f70

• Coreutils (8.31) - 5,284 KB:
Домашняя страница: *https://www.gnu.org/software/coreutils/*
Ссылка на загрузку: *https://ftp.gnu.org/gnu/coreutils/coreutils-8.31.tar.xz*
Контрольная сумма MD5: 0009a224d8e288e8ec406ef0161f9293

• DejaGNU (1.6.2) - 514 KB:
Домашняя страница: *https://www.gnu.org/software/dejagnu/*
Ссылка на загрузку: *https://ftp.gnu.org/gnu/dejagnu/dejagnu-1.6.2.tar.gz*
Контрольная сумма MD5: e1b07516533f351b3aba3423fafeffd6

• Diffutils (3.7) - 1,415 KB:
Домашняя страница: *https://www.gnu.org/software/diffutils/*
Ссылка на загрузку: *https://ftp.gnu.org/gnu/diffutils/diffutils-3.7.tar.xz*
Контрольная сумма MD5: 4824adc0e95dbbf11dfbdfaad6a1e461

• E2fsprogs (1.45.3) - 7,741 KB:
Домашняя страница: *http://e2fsprogs.sourceforge.net/*
Ссылка на загрузку: *https://downloads.sourceforge.net/project/e2fsprogs/e2fsprogs/v1.45.3/e2fsprogs-1.45.3.tar.gz*
Контрольная сумма MD5: 447a225c05f0a81121f6ddffbf55b06c

• Elfutils (0.177) - 8,645 KB:
Домашняя страница: *https://sourceware.org/ftp/elfutils/*
Ссылка на загрузку: *https://sourceware.org/ftp/elfutils/0.177/elfutils-0.177.tar.bz2*
Контрольная сумма MD5: 0b583722f911e1632544718d502aab87

• Eudev (3.2.8) - 1,850 KB:
Ссылка на загрузку: *https://dev.gentoo.org/~blueness/eudev/eudev-3.2.8.tar.gz*
Контрольная сумма MD5: ce166b3fdd910c2a4a840378f48fedaf

• Expat (2.2.7) - 415 KB:
Домашняя страница: *https://libexpat.github.io/*
Ссылка на загрузку: *https://prdownloads.sourceforge.net/expat/expat-2.2.7.tar.xz*
Контрольная сумма MD5: 3659bc0938db78815b5f5a9c24d732aa

• 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 (5.37) - 867 KB:
Домашняя страница: *https://www.darwinsys.com/file/*
Ссылка на загрузку: *ftp://ftp.astron.com/pub/file/file-5.37.tar.gz*
Контрольная сумма MD5: 80c29aca745466c6c24d11f059329075

Note
File (5.37) может быть недоступен в указанном месте. Администраторы сайта иногда удаляют старые версии при выпуске новых. Альтернативное место загрузки, которое может иметь нужную версию, можно найти по адресу: *http://www.linuxfromscratch.org/lfs/download.html#ftp*.

• Findutils (4.6.0) - 3,692 KB:
Домашняя страница: *https://www.gnu.org/software/findutils/*
Ссылка на загрузку: *https://ftp.gnu.org/gnu/findutils/findutils-4.6.0.tar.gz*
Контрольная сумма MD5: 9936aa8009438ce185bea2694a997fc1

• 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

• Gawk (5.0.1) - 3,063 KB:
Домашняя страница: *https://www.gnu.org/software/gawk/*
Ссылка на загрузку: *https://ftp.gnu.org/gnu/gawk/gawk-5.0.1.tar.xz*
Контрольная сумма MD5: f9db3f6715207c6f13719713abc9c707

• GCC (9.2.0) - 68,953 KB:
Домашняя страница: *https://gcc.gnu.org/*
Ссылка на загрузку: *https://ftp.gnu.org/gnu/gcc/gcc-9.2.0/gcc-9.2.0.tar.xz*
Контрольная сумма MD5: 3818ad8600447f05349098232c2ddc78

• GDBM (1.18.1) - 920 KB:
Домашняя страница: *https://www.gnu.org/software/gdbm/*
Ссылка на загрузку: *https://ftp.gnu.org/gnu/gdbm/gdbm-1.18.1.tar.gz*
Контрольная сумма MD5: 988dc82182121c7570e0cb8b4fcd5415

• Gettext (0.20.1) - 9,128 KB:
Домашняя страница: *https://www.gnu.org/software/gettext/*
Ссылка на загрузку: *https://ftp.gnu.org/gnu/gettext/gettext-0.20.1.tar.xz*
Контрольная сумма MD5: 9ed9e26ab613b668e0026222a9c23639

• Glibc (2.30) - 16,189 KB:
Домашняя страница: *https://www.gnu.org/software/libc/*
Ссылка на загрузку: *https://ftp.gnu.org/gnu/glibc/glibc-2.30.tar.xz*
Контрольная сумма MD5: 2b1dbdf27b28620752956c061d62f60c

• GMP (6.1.2) - 1,901 KB:
Домашняя страница: *https://www.gnu.org/software/gmp/*
Ссылка на загрузку: *https://ftp.gnu.org/gnu/gmp/gmp-6.1.2.tar.xz*
Контрольная сумма MD5: f58fa8001d60c4c77595fbbb62b63c1d

• 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 (3.3) - 1,440 KB:
Домашняя страница: *https://www.gnu.org/software/grep/*
Ссылка на загрузку: *https://ftp.gnu.org/gnu/grep/grep-3.3.tar.xz*
Контрольная сумма MD5: 05d0718a1b7cc706a4bdf8115363f1ed

• Groff (1.22.4) - 4,044 KB:
Домашняя страница: *https://www.gnu.org/software/groff/*
Ссылка на загрузку: *https://ftp.gnu.org/gnu/groff/groff-1.22.4.tar.gz*
Контрольная сумма MD5: 08fb04335e2f5e73f23ea4c3adbf0c5f

• GRUB (2.04) - 6,245 KB:
Домашняя страница: *https://www.gnu.org/software/grub/*
Ссылка на загрузку: *https://ftp.gnu.org/gnu/grub/grub-2.04.tar.xz*
Контрольная сумма MD5: 5aaca6713b47ca2456d8324a58755ac7

• Gzip (1.10) - 757 KB:
Домашняя страница: *https://www.gnu.org/software/gzip/*
Ссылка на загрузку: *https://ftp.gnu.org/gnu/gzip/gzip-1.10.tar.xz*
Контрольная сумма MD5: 691b1221694c3394f1c537df4eee39d3

• Iana-Etc (2.30) - 201 KB:
Домашняя страница: *http://freecode.com/projects/iana-etc*
Ссылка на загрузку: *http://anduin.linuxfromscratch.org/LFS/iana-etc-2.30.tar.bz2*
Контрольная сумма MD5: 3ba3afb1d1b261383d247f46cb135ee8

• Inetutils (1.9.4) - 1,333 KB:
Домашняя страница: *https://www.gnu.org/software/inetutils/*
Ссылка на загрузку: *https://ftp.gnu.org/gnu/inetutils/inetutils-1.9.4.tar.xz*
Контрольная сумма MD5: 87fef1fa3f603aef11c41dcc097af75e

• 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 (5.2.0) - 713 KB:
Домашняя страница: *https://www.kernel.org/pub/linux/utils/net/iproute2/*
Ссылка на загрузку: *https://www.kernel.org/pub/linux/utils/net/iproute2/iproute2-5.2.0.tar.xz*
Контрольная сумма MD5: 0cb2736e7bc2f56254a363d3d23703b7

• Kbd (2.2.0) - 1,090 KB:
Домашняя страница: *http://ftp.altlinux.org/pub/people/legion/kbd*
Ссылка на загрузку: *https://www.kernel.org/pub/linux/utils/kbd/kbd-2.2.0.tar.xz*
Контрольная сумма MD5: d1d7ae0b5fb875dc082731e09cd0c8bc

• Kmod (26) - 540 KB:
Ссылка на загрузку: *https://www.kernel.org/pub/linux/utils/kernel/kmod/kmod-26.tar.xz*
Контрольная сумма MD5: 1129c243199bdd7db01b55a61aa19601

• Less (551) - 339 KB:
Домашняя страница: *http://www.greenwoodsoftware.com/less/*
Ссылка на загрузку: *http://www.greenwoodsoftware.com/less/less-551.tar.gz*
Контрольная сумма MD5: 4ad4408b06d7a6626a055cb453f36819

• LFS-Bootscripts (20190524) - 32 KB:
Ссылка на загрузку: *https://linuxfromscratch.ru/lfs/downloads/9.0/lfs-bootscripts-20190524.tar.xz*
Контрольная сумма MD5: c91b11e366649c9cec60c2552820fed5

• Libcap (2.27) - 67 KB:
Домашняя страница: *https://sites.google.com/site/fullycapable/*
Ссылка на загрузку: *https://www.kernel.org/pub/linux/libs/security/linux-privs/libcap2/libcap-2.27.tar.xz*
Контрольная сумма MD5: 2e8f9fab32eb5ccb37969fe317fd17aa

• Libffi (3.2.1) - 920 KB:
Домашняя страница: *https://sourceware.org/libffi/*
Ссылка на загрузку: *ftp://sourceware.org/pub/libffi/libffi-3.2.1.tar.gz*
Контрольная сумма MD5: 83b89587607e3eb65c70d361f13bab43

• Libpipeline (1.5.1) - 965 KB:
Домашняя страница: *http://libpipeline.nongnu.org/*
Ссылка на загрузку: *https://download.savannah.gnu.org//releases/libpipeline/libpipeline-1.5.1.tar.gz*
Контрольная сумма MD5: 4c8fe6cd85422baafd6e060f896c61bc

• Libtool (2.4.6) - 951 KB:
Домашняя страница: *https://www.gnu.org/software/libtool/*
Ссылка на загрузку: *https://ftp.gnu.org/gnu/libtool/libtool-2.4.6.tar.xz*
Контрольная сумма MD5: 1bfb9b923f2c1339b4d2ce1807064aa5

• Linux (5.2.8) - 104,555 KB:
Домашняя страница: *https://www.kernel.org/*
Ссылка на загрузку: *https://www.kernel.org/pub/linux/kernel/v5.x/linux-5.2.8.tar.xz*
Контрольная сумма MD5: 602dd0ecb8646e539fefb2beb6eb6fe0

Note
Ядро Linux обновляется достаточно часто, по причине разрешения проблем безопасности. Необходимо использовать последний стабильный релиз 5.2.x версии ядра, если в разделе опечаток и неточностей не указано иное.
Для пользователей, у которых ограниченный или тарифицируемый выход в сеть интернет, и который хотят обновить ядро Linux, можно скачать базовую версию ядра а затем применить к ней патчи, которые могут быть загружены отдельно. Обновление до последней версии при помощи патчей может сэкономить время и стоимость.

• M4 (1.4.18) - 1,180 KB:
Домашняя страница: *https://www.gnu.org/software/m4/*
Ссылка на загрузку: *https://ftp.gnu.org/gnu/m4/m4-1.4.18.tar.xz*
Контрольная сумма MD5: 730bb15d96fffe47e148d1e09235af82

• Make (4.2.1) - 1,932 KB:
Домашняя страница: *https://www.gnu.org/software/make/*
Ссылка на загрузку: *https://ftp.gnu.org/gnu/make/make-4.2.1.tar.gz*
Контрольная сумма MD5: 7d0dcb6c474b258aab4d54098f2cf5a7

• Man-DB (2.8.6.1) - 1,787 KB:
Домашняя страница: *https://www.nongnu.org/man-db/*
Ссылка на загрузку: *https://download.savannah.gnu.org//releases/man-db/man-db-2.8.6.1.tar.xz*
Контрольная сумма MD5: 22e82fe1127f4ca95de7100168a927d1

• Man-pages (5.02) - 1,630 KB:
Домашняя страница: *https://www.kernel.org/doc/man-pages/*
Ссылка на загрузку: *https://www.kernel.org/pub/linux/docs/man-pages/man-pages-5.02.tar.xz*
Контрольная сумма MD5: 136e5e3380963571a079693d8ae38f52

• Meson (0.51.1) - 1,418 KB:
Домашняя страница: *https://mesonbuild.com*
Ссылка на загрузку: *https://github.com/mesonbuild/meson/releases/download/0.51.1/meson-0.51.1.tar.gz*
Контрольная сумма MD5: 48787e391ec5c052799a3dd491f73909

• MPC (1.1.0) - 685 KB:
Домашняя страница: *http://www.multiprecision.org/*
Ссылка на загрузку: *https://ftp.gnu.org/gnu/mpc/mpc-1.1.0.tar.gz*
Контрольная сумма MD5: 4125404e41e482ec68282a2e687f6c73

• MPFR (4.0.2) - 1,409 KB:
Домашняя страница: *https://www.mpfr.org/*
Ссылка на загрузку: *http://www.mpfr.org/mpfr-4.0.2/mpfr-4.0.2.tar.xz*
Контрольная сумма MD5: 320fbc4463d4c8cb1e566929d8adc4f8

• Ninja (1.9.0) - 187 KB:
Домашняя страница: *https://ninja-build.org/*
Ссылка на загрузку: *https://github.com/ninja-build/ninja/archive/v1.9.0/ninja-1.9.0.tar.gz*
Контрольная сумма MD5: f340be768a76724b83e6daab69009902

• Ncurses (6.1) - 3,288 KB:
Домашняя страница: *https://www.gnu.org/software/ncurses/*
Ссылка на загрузку: *https://ftp.gnu.org/gnu/ncurses/ncurses-6.1.tar.gz*
Контрольная сумма MD5: 98c889aaf8d23910d2b92d65be2e737a

• OpenSSL (1.1.1c) - 8,657 KB:
Домашняя страница: *https://www.openssl.org/*
Ссылка на загрузку: *https://www.openssl.org/source/openssl-1.1.1c.tar.gz*
Контрольная сумма MD5: 15e21da6efe8aa0e0768ffd8cd37a5f6

• 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 (5.30.0) - 12,129 KB:
Домашняя страница: *https://www.perl.org/*
Ссылка на загрузку: *https://www.cpan.org/src/5.0/perl-5.30.0.tar.xz*
Контрольная сумма MD5: 037c35000550bdcb47552ad0f6d3064d

• Pkg-config (0.29.2) - 1,970 KB:
Домашняя страница: *https://www.freedesktop.org/wiki/Software/pkg-config*
Ссылка на загрузку: *https://pkg-config.freedesktop.org/releases/pkg-config-0.29.2.tar.gz*
Контрольная сумма MD5: f6e931e319531b736fadc017f470e68a

• Procps (3.3.15) - 884 KB:
Домашняя страница: *https://sourceforge.net/projects/procps-ng*
Ссылка на загрузку: *https://sourceforge.net/projects/procps-ng/files/Production/procps-ng-3.3.15.tar.xz*
Контрольная сумма MD5: 2b0717a7cb474b3d6dfdeedfbad2eccc

• Psmisc (23.2) - 297 KB:
Домашняя страница: *http://psmisc.sourceforge.net/*
Ссылка на загрузку: *https://sourceforge.net/projects/psmisc/files/psmisc/psmisc-23.2.tar.xz*
Контрольная сумма MD5: 0524258861f00be1a02d27d39d8e5e62

• Python (3.7.4) - 16,730 KB:
Домашняя страница: *https://www.python.org/*
Ссылка на загрузку: *https://www.python.org/ftp/python/3.7.4/Python-3.7.4.tar.xz*
Контрольная сумма MD5: d33e4aae66097051c2eca45ee3604803

• Python Documentation (3.7.4) - 6,068 KB:
Ссылка на загрузку: *https://docs.python.org/ftp/python/doc/3.7.4/python-3.7.4-docs-html.tar.bz2*
Контрольная сумма MD5: c410337e954dbba2d04fe169c355a6a2

• Readline (8.0) - 2,907 KB:
Домашняя страница: *https://tiswww.case.edu/php/chet/readline/rltop.html*
Ссылка на загрузку: *https://ftp.gnu.org/gnu/readline/readline-8.0.tar.gz*
Контрольная сумма MD5: 7e6c1f16aee3244a69aba6e438295ca3

• Sed (4.7) - 1,268 KB:
Домашняя страница: *https://www.gnu.org/software/sed/*
Ссылка на загрузку: *https://ftp.gnu.org/gnu/sed/sed-4.7.tar.xz*
Контрольная сумма MD5: 777ddfd9d71dd06711fe91f0925e1573

• Shadow (4.7) - 1,587 KB:
Ссылка на загрузку: *https://github.com/shadow-maint/shadow/releases/download/4.7/shadow-4.7.tar.xz*
Контрольная сумма MD5: f7ce18c8dfd05f1a009266cb604d58b7

• Sysklogd (1.5.1) - 88 KB:
Домашняя страница: *http://www.infodrom.org/projects/sysklogd/*
Ссылка на загрузку: *http://www.infodrom.org/projects/sysklogd/download/sysklogd-1.5.1.tar.gz*
Контрольная сумма MD5: c70599ab0d037fde724f7210c2c8d7f8

• Sysvinit (2.95) - 122 KB:
Домашняя страница: *https://savannah.nongnu.org/projects/sysvinit*
Ссылка на загрузку: *https://download.savannah.gnu.org//releases/sysvinit/sysvinit-2.95.tar.xz*
Контрольная сумма MD5: 66f488c952c70ff4245774fc7e87e529

• Tar (1.32) - 2,055 KB:
Домашняя страница: *https://www.gnu.org/software/tar/*
Ссылка на загрузку: *https://ftp.gnu.org/gnu/tar/tar-1.32.tar.xz*
Контрольная сумма MD5: 83e38700a80a26e30b2df054e69956e5

• Tcl (8.6.9) - 9,772 KB:
Домашняя страница: *http://tcl.sourceforge.net/*
Ссылка на загрузку: *https://downloads.sourceforge.net/tcl/tcl8.6.9-src.tar.gz*
Контрольная сумма MD5: aa0a121d95a0e7b73a036f26028538d4

• Texinfo (6.6) - 4,831 KB:
Домашняя страница: *https://www.gnu.org/software/texinfo/*
Ссылка на загрузку: *https://ftp.gnu.org/gnu/texinfo/texinfo-6.6.tar.xz*
Контрольная сумма MD5: 5231da3e6aa106cd0532b8609e5b3702

• Time Zone Data (2019b) - 376 KB:
Домашняя страница: *https://www.iana.org/time-zones*
Ссылка на загрузку: *https://www.iana.org/time-zones/repository/releases/tzdata2019b.tar.gz*
Контрольная сумма MD5: b26b5d7d844cb96c73ed2fb6d588daaf

• Udev-lfs Tarball (udev-lfs-20171102) - 11 KB:
Ссылка на загрузку: *http://anduin.linuxfromscratch.org/LFS/udev-lfs-20171102.tar.xz*
Контрольная сумма MD5: 27cd82f9a61422e186b9d6759ddf1634

• Util-linux (2.34) - 4,859 KB:
Домашняя страница: *http://freecode.com/projects/util-linux*
Ссылка на загрузку: *https://www.kernel.org/pub/linux/utils/util-linux/v2.34/util-linux-2.34.tar.xz*
Контрольная сумма MD5: a78cbeaed9c39094b96a48ba8f891d50

• Vim (8.1.1846) - 14,078 KB:
Домашняя страница: *https://www.vim.org*
Ссылка на загрузку: *https://github.com/vim/vim/archive/v8.1.1846/vim-8.1.1846.tar.gz*
Контрольная сумма MD5: 4f129a05254d93c739fcede843df87df

• XML::Parser (2.44) - 232 KB:
Домашняя страница: *https://github.com/chorny/XML-Parser*
Ссылка на загрузку: *https://cpan.metacpan.org/authors/id/T/TO/TODDR/XML-Parser-2.44.tar.gz*
Контрольная сумма MD5: af4813fe3952362451201ced6fbce379

• Xz Utils (5.2.4) - 1030 KB:
Домашняя страница: *https://tukaani.org/xz*
Ссылка на загрузку: *https://tukaani.org/xz/xz-5.2.4.tar.xz*
Контрольная сумма MD5: 003e4d0b1b1899fc6e3000b24feddf7c

• Zlib (1.2.11) - 457 KB:
Домашняя страница: *https://www.zlib.net/*
Ссылка на загрузку: *https://zlib.net/zlib-1.2.11.tar.xz*
Контрольная сумма MD5: 85adef240c5f370b308da8c938951a68

Общий размер этих пакетов: примерно 391 MB

3.3. Необходимые патчи

К пакетам потребуются патчи. Они исправляют ошибки в пакетах которые должны быть исправлены владельцем пакета. Патчи также вносят небольшие модификации чтобы облегчить работу с пакетами. Следующие патчи необходимы для сборки системы LFS:

• Bzip2 Documentation Patch - 1.6 KB:
Ссылка на загрузку: *https://linuxfromscratch.ru/patches/lfs/9.0/bzip2-1.0.8-install_docs-1.patch*
Контрольная сумма MD5: 6a5ac7e89b791aae556de0f745916f7f

• Coreutils Internationalization Fixes Patch - 168 KB:
Ссылка на загрузку: *https://linuxfromscratch.ru/patches/lfs/9.0/coreutils-8.31-i18n-1.patch*
Контрольная сумма MD5: a9404fb575dfd5514f3c8f4120f9ca7d

• Glibc FHS Patch - 2.8 KB:
Ссылка на загрузку: *https://linuxfromscratch.ru/patches/lfs/9.0/glibc-2.30-fhs-1.patch*
Контрольная сумма MD5: 9a5997c3452909b1769918c759eff8a2

• Kbd Backspace/Delete Fix Patch - 12 KB:
Ссылка на загрузку: *https://linuxfromscratch.ru/patches/lfs/9.0/kbd-2.2.0-backspace-1.patch*
Контрольная сумма MD5: f75cca16a38da6caa7d52151f7136895

• Sysvinit Consolidated Patch - 2.4 KB:
Ссылка на загрузку: *https://linuxfromscratch.ru/patches/lfs/9.0/sysvinit-2.95-consolidated-1.patch*
Контрольная сумма MD5: 4900322141d493e74020c9cf437b2cdc

Размер патчей примерно: 186.8 KB

В дополнение к необходимым патчам, существует множество опциональных, созданных сообществом LFS. Эти патчи исправляют проблемы или добавляют функциональность которая не включена по умолчанию. Не забудьте ознакомиться с базой патчей, расположенной по адресу *https://linuxfromscratch.ru/patches/downloads/* и воспользуйтесь ими в соответствии с требованиями к вашей системе.

Chapter 4. Заключительные этапы подготовки

4.1. Введение

В этой главе будет выполнено несколько задач для подготовки к сборки временной системы. Мы создадим каталог в $LFS для установки временного набора инструментов, и добавим непривилегированного пользователя, чтобы снизить риск, и настроим окружение для этого пользователя. Также будут даны объяснения по поводу стандартной единицы времени сборки (SBU), для того, чтобы была возможность оценить приблизительное время сборки каждого пакета. Также, будет предоставлена информация по поводу набора тестов для пакетов.

4.2. Создание каталога $LFS/tools

Все программы которые будут скомпилированы в главе Глава 5 будут установлены в каталог `$LFS/tools` чтобы можно было оставить их отдельно от сборки конечной системы в главе Chapter 6. Программы которые будут скомпилированы - временные инструменты и не будут входить в конечную сборку LFS системы. После использования временных инструментов, от них можно избавиться. Использование каталога $LFS/tools для хранения временного набора инструментов также полезно для того, чтобы не засорять рабочие каталоги хост-системы (что можно случайно сделать в Глава 5).

Создайте каталог, выполнив команду от пользователя `root`:

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

Следующим шагом, будет создание символической ссылки `/tools` в хост-системе, которая будет указывать на созданный каталог с временными инструментами. Запустите команду от пользователя `root`:

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

Note

Вышеприведенная команда верна. Команда **ln** имеет несколько синтаксических вариаций, поэтому обязательно проверьте **info coreutils ln** and `ln(1)` перед тем, как сообщить о предполагаемой ошибке.

Созданная символическая ссылка позволяет выполнить компиляцию временного набора инструментов по ссылке `/tools` , это означает что компилятор, ассемблер и компоновщик смогут работать как в главе 5 (на тот момент, когда мы все еще используем некоторые инструменты хост-системы) и в дальнейшем (когда мы выполним команду “chroot” в раздел LFS).

4.3. Создание пользователя LFS

Когда мы находимся в системе под пользователем `root`, одна единственная ошибка может привести к повреждению или поломке хост-системы. Следовательно мы рекомендуем выполнять сборку пакетов для временного набора инструментов в этой главе от обычного пользователя, без привилегий. Вы можете использовать произвольного пользователя, но для

упрощения настройки чистого рабочего окружения создайте нового пользователя с именем `lfs` как члена группы `lfs` и используйте этого пользователя на время всего процесса установки временного набора инструментов. Выполните команду от пользователя `root`:

```
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`
Опция добавляет пользователя `lfs` в созданную 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`), изменив местоположение ввода на специальное null устройство.

`lfs`
Это наименование созданного пользователя и группы.

Чтобы войти в систему как пользователь lfs (в отличие от смены пользователя `lfs` когда вы авторизованы как пользователь `root`, для которого не требуется , чтобы пользователь `lfs` имел пароль), создадим для учётной записи пароль:

```
passwd lfs
```

Предоставим `lfs` полный доступ к `$LFS/tools` сделав пользователя `lfs` владельцем каталога:

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

Если был создан отдельный рабочий каталог, как предлагается, предоставьте пользователю `lfs` права на следующий каталог:

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

Далее, выполним вход как пользователь `lfs`. Это действие можно выполнить в графической оболочке, используя виртуальный терминал, или в обычной пользовательской среде:

```
su - lfs
```

Аргумент “-” предает значение для команды **su** для начала запуска login-оболочки, в отличие от обычной. Различия между двумя типами оболочек можно подробно изучить в `bash(1)` и **информация о bash**.

4.4. Настройка окружения

Настроим хорошо работающее окружение. Создадим два файла для оболочки **bash**. При входе в систему как пользователь `lfs` выполните следующую команду для создания нового файла `.bash_profile` :

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

Когда выполнен вход под пользователем `lfs`, при инициализации оболочки *login*, будет читать данные из файла `/etc/profile` хост-системы (который возможно будет содержать некоторые настройки и дополнительные переменные), и затем данные из файла `.bash_profile` . Команда **exec env -i../bin/bash** в файле `.bash_profile` заменяет запущенную оболочку новой полностью пустой средой, кроме таких переменных, как HOME, TERM, и PS1. Такой подход гарантирует, что нежелательные и потенциально опасные переменные окружения хост-системы не попадут в среду сборки. В результате чего мы получаем чистое окружение.

Новый экземпляр оболочки - это *non-login* оболочка, которая не выполняет чтение файла `/etc/profile` или `.bash_profile` , но выполняет чтение из файла `.bashrc` . Создадим файл `.bashrc` :

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

Команда **set +h** отключает хеш-функцию **bash**. Хеширование полезно когда **bash** использует хеш-таблицу для хранения полного пути исполняемых файлов чтобы избежать поиска в переменной окружения PATH при каждом обращении. Однако новые инструменты должны быть использованы сразу после их установки. При выключении хеш-функции оболочка будет всегда искать переменную окружения PATH когда программу необходимо запустить. Таким образом, оболочка будет выполнять поиск скомпилированных инструментов в каталоге `$LFS/tools` , как только они станут доступны без запоминания предыдущей версии той же программы в другом расположении.

Указание для значения 022 маски создания файлов пользователя (umask) для создания новых файлов и каталогов позволяет выполнять запись только их владельцу, но они останутся доступными на чтение и выполнение для остальных пользователей (при условии, что режимы по умолчанию используют системный вызов `open(2)`, новые файлы получат разрешение 644 а каталоги 755).

Переменная окружения LFS должны указывать на выбранную точку монтирования.

Переменная LC_ALL управляет локализацией некоторых программ, и формирует сообщения в соответствии с соглашениями локализаций указанной страны. Если указать значение LC_ALL в “POSIX” или “C” (что одно и тоже) гарантирует, что все будет работает так, как ожидается в chroot окружении.

Переменная LFS_TGT содержит совместимое описание компьютера, которая будет использоваться при сборке кросс-компилятора и компоновщика и затем при кросс-компиляции временного набора инструментов. Более подробная информация содержится в разделе Section 5.2, “Технические примечания относительно временного набора инструментов”.

Указание значения /tools/bin перед стандартным содержимым переменной PATH дает возможность оболочке сразу использовать программы и библиотеки, которые были установлены в этот каталог. В сочетании с отключением хеширования, снижается риск, что старые программы будут использованы хост-системой когда те же программы будут доступны в среде окружения, в главе 5.

Наконец, чтобы полностью подготовить среду для временных инструментов, укажите источник только что созданного профиля пользователя:

```
source ~/.bash_profile
```

4.5. Информация о SBU (Стандартная единица времени сборки)

Многим хотелось бы знать заранее приблизительное время, которое будет затрачено на компиляцию и установку каждого пакета. Так как система LFS может быть создана на разных машинах, точное время не оценить. На компиляцию и сборку одного из больших пакетов - Glibc, может уйти примерно 20 минут на быстрых компьютерах, но на медленных машинах может занять до трёх дней! Вместо фактического времени, будет использоваться показатель SBU (Стандартная единица времени сборки).

Показатель SBU вычисляется следующим образом: первый пакет, который будет скомпилирован в этой книге - является Binutils, сборка которого обсуждается в разделе Глава 5. Время, которое будет затрачено на компиляцию этого пакета будет определено как одна единица времени сборки (Standard Build Unit) или (SBU). Все остальные значения времени при компиляции других пакетов будет считаться относительно этой единицы.

Например, рассмотрим пакет, время сборки которого равно 4.5 SBU. Это означает, если сборка и установка пакета Binutils заняла 10 минут, то время на сборки и установки будет приблизительно 45 минут. К счастью, время сборки многих пакетов гораздо меньше, чем у Binutils.

В целом, информация о SBU может быть не точна и значения могут различаться в пределах нескольких десятков минут. Потому что существует множество факторов, таки как версия GCC хост-системы и прочие аспекты.

Note

Для многих современных компьютеров, с несколькими процессорами или ядрами, время компиляции пакета можно сократить, выполняя процедуру "параллельной сборки", указанием в переменной окружения или аргументом при использовании программы **make** сколько процессоров доступно. Например процессор Core2Duo может выполнять сборку в два потока, если указать следующее:

```
export MAKEFLAGS='-j 2'
```

или выполнить компиляции с указанием:

```
make -j2
```

Когда на машине несколько процессоров, информация SBU будет искажена больше, чем при выполнении сборки в один поток. В некоторых случаях процесс сборки может завершиться неудачно. К тому же анализ выходного потока при совершении сборки будет трудночитаем, так как вывод будет содержать строки вперемешку с разных потоков. Если произошла проблема сборки в несколько потоков, выполните сборку в один поток, и проанализируйте сообщение об ошибке.

4.6. О тестировании

Многие пакеты содержат инструменты для тестирования. Выполнить наборы тестов для скомпилированного пакета может быть хорошей идеей, чтобы проверить работоспособность и убедиться что все работает так, как предполагалось разработчиками пакета. Однако это не гарантирует что пакет не содержит ошибок.

Некоторые наборы тестов важнее, чем остальные. Например наборы тестов для главных пакетов временного инструментария - —GCC, Binutils, и Glibc— чрезвычайно важны и необходимы, для правильной работы всей системы. Наборы тестов для пакетов GCC и Glibc могуть занять очень много времени, особенно на слабом оборудовании. Но их выполнение настоятельно рекомендуется.

Note

Опыт показывает, что выполнение тестов пакетов, в главе Глава 5 не будет содержательным. Хост-система в любом случае будет оказывать влияние на выполнение тестов, и может приводить к непрогнозируемым результатам, к тому же инструменты созданные в главе Глава 5 временные и потом будет исключены. Мы не рекомендуем выполнять тесты в этой главе для рядового читателя. Инструкции по тестированию в основном представлены для тестировщиков и разработчиков и не обязательны.

Общая проблема при тестировании пакетов Binutils и GCC связана я псевдотерминалами (PTYs). И может выдавать большое количество неудачно завершенных тестов. Причин может быть множество, но скорее всего причина в том, в хост-системе не правильно настроена файловая система `devpts`. Этот вопрос обсуждается более подробно на этой странице: *https://linuxfromscratch.ru/lfs/faq.html#no-ptys*.

Иногда при выполнении тестирования пакетов результат будет неудачным, но разработчики могут считать такую ситуацию не критичной. Ознакомьтесь с отчётами, расположенными по ссылке *https://linuxfromscratch.ru/lfs/build-logs/9.0/* чтобы проверить являются ли проблемы ожидаемыми. В отчёте представлены результаты тестирования всех пакетов книги.

Chapter 5. Сборка временной системы

5.1. Введение

Это глава описывает процесс создания минимальной Linux системы. Эта система будет содержать только те инструменты, которые будут нужны для сборки конечной системы LFS в главе Chapter 6 и позволит работать с удобными настройками, чем обычная минимальная среда.

Необходимо выполнить два шага для сборки минимальной системы. Первым шагом будет выполнена сборка независимого от хост системы набора инструментов (компилятор, ассемблер, компоновщик, библиотеки, и ещё несколько полезных утилит). Вторым шагом, с использованием этих инструментов будут скомпилированы остальные необходимые пакеты.

Все пакеты, которые будут скомпилированы в этой главе, будут установлены в каталог `$LFS/tools` чтобы хранить их отдельно от хост системы и конечной системы LFS, которая будет создана в следующей главе. Как обсуждалось ранее, это набор инструментов - временный, и нам не требуется его наличие в системе LFS.

5.2. Технические примечания относительно временного набора инструментов

В этом разделе объясняются детали по всему процессу сборки. Не обязательно сразу понимать все что здесь написано. Большая часть информации станет понятной в процессе сборки. К этому разделу можно обратиться в любое время.

Главной задачей главы Глава 5 является создание временной области которая содержит требуемый набор инструментов, который может быть изолирован от хост-системы. Использование команды **chroot** в оставшихся главах обеспечит чистое окружение и беспроблемную сборку конечной системы LFS. Такой процесс был выбран чтобы минимизировать риски для новых читателей и одновременно как образовательная цель.

Note

Перед тем как приступить, необходимо знать название платформы хост-системы, часто называемой целевой "триплет". Простой способ это определить - выполнить команду **config.guess** которая поставляется с исходными кодами многих пакетов. Распакуйте пакет Binutils и выполните команду **`./config.guess`** и запишите вывод. Например, для 32 битного процессора intel вывод будет *i686-pc-linux-gnu*. Для 64 битного - *x86_64-pc-linux-gnu*.

Также, необходим знать название динамического компоновщика, часто называемого динамический загрузчик (не путать со стандартным компоновщиком **ld**, поставляемый в пакете Binutils). Динамический компоновщик поставляется в пакете Glibc. Он находит и загружает общие библиотеки необходимые для выполнения и запуска программ, после чего запускает программу. Название динамического компоновщика на 32-битной машине с процессором intel будет `ld-linux.so.2` (`ld-linux-x86-64.so.2` для 64-битных систем). Надежный способ узнать название динамического компоновщика на хост-системе - проверить произвольный бинарный файл хост-системы выполнив команду **`readelf -l <название бинарного файла> | grep interpreter`** и проверить вывод. Официальная ссылка, охватывающая все платформы, находится в файле `shlib-versions` в корне каталога с исходными кодом пакета Glibc.

Ключевые технические указания как в Глава 5 работает метод сборки:

- Выполнив настройку названия платформы хост-системы, изменив поле "vendor" целевого триплета с помощью переменной окружения `LFS_TGT`, чтобы выполнить первую сборку пакетов Binutils и GCC с совместимым кросс-компоновщиком и кросс-компилятором. Вместо того, чтобы создать файлы для другой архитектуры, кросс-компоновщик и кросс-компилятор создадут файлы совместимые с текущим аппаратным обеспечением.
- Временные библиотеки будут скомпилированы кросс-компилятором, потому что он не зависит от хост-системы . Такой метод исключает потенциальное загрязнение целевой системы, уменьшая вероятность того, что будут включены какие либо заголовочные файлы и библиотеки хост-системы в временный набор инструментов. Кросс компиляция также позволяет выполнять сборку как 32-битных, так и 64-битных библиотек на 64-битном совместимом аппаратном обеспечении.
- Аккуратная манипуляция с исходными кодами пакета GCC будет указывать компилятору какой целевой динамический загрузчик необходимо использовать.

Пакет Binutils устанавливается первым, потому что команда **configure** запускает GCC и Glibc и выполняет различные функциональные тесты ассемблера и компоновщика для определения какие функции программного обеспечения включить или выключить. Некорректно настроенный GCC или Glibc могут привести к неработоспособности всего временного набора инструментов, и такая проблема может проявиться только в конце сборки дистрибутива. Сбои при выполнении тестов могут сигнализировать о наличии таких ошибок.

Пакет Binutils выполняет установку нового ассемблера и компоновщика в два каталога: `/tools/bin` и `/tools/$LFS_TGT/bin`. Инструменты в одном каталоге жестко связаны. Важной особенностью компоновщика является порядок поиска библиотек. Детальную информацию можно получить используя команду **ld** с аргументом **`ld --verbose | grep`**

SEARCH. Например команда **ld --verbose | grep SEARCH** покажет текущие пути поиска и их порядок. Она показывает, какие файлы связаны **ld** при компиляции проверочной программы и указание аргумента *--verbose* компоновщику. Например команда **gcc dummy.c -Wl,--verbose 2>&1 | grep succeeded** покажет что все файлы были успешно открыты в процессе линковки.

Следующий пакет который будет установлен - GCC. Пример вывода в процессе выполнения команды **configure** будет:

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

Это важный момент, по причинам указанным выше. Также демонстрируется что сценарий конфигурирования GCC не выполнят поиск по каталогам переменной окружения PATH для поиска инструментов. Однаком во время самой работы **gcc** необязательно, что одни и те же пути будет использованы. Чтобы узнать, какой стандартный компоновщик **gcc** будет использовать, запустите команду: **gcc -print-prog-name=ld**.

Подробная информация может быть получена в**gcc** указанием аргумента *-v* при компиляции проверочной программы. Например команда **gcc -v dummy.c** покажет подробную информацию о препроцессоре, компиляции и стадий сборки, включая пути поиска **gcc** и их порядок.

Далее будут установлены изолированные заголовочные файлы Linux API (Linux API headers). Это позволит библиотеке C (Glibc) взаимодействовать с функциями, которые предоставляет ядро Linux.

Следующий пакет который будет установлен - Glibc. Необходимые компоненты для сборки Glibc - это компилятор, инструменты для работы с бинарными файлами, и заголовочные файлы ядра Linux (Linux API headers). Проблем с компилятором, как правило, не должно быть. Поскольку Glibc будет всегда использовать компилятор ссылающийся на аргумент *--host* который будет указан программе **configure**; например, в нашем случае, компилятором будет **i686-lfs-linux-gnu-gcc**.С инструментами для работы с бинарными файлами, и заголовочными файлы ядра Linux (Linux API headers) все будет немного сложнее. Поэтому не рискуйте и используйте доступные опции конфигурации чтобы быть абсолютно уверенным в корректности. После запуска программы **configure** проверьте содержимое файла `config.make` в каталоге `glibc-build` . Убедитесь, что используется *CC="i686-lfs-gnu-gcc"* для контроля, каким инструментом для работы с бинарными файлами и используются ли аргументы *-nostdinc* и *-isystem* для контроля какие пути поиска будет использовать компилятор. Эти пункты подчеркивают важный аспект пакета Glibc—он очень самодостаточен с точки зрения его механизма сборки и обычно не полагается на значения по умолчанию для временного набора инструментов.

Во время второго прохода сборки пакета Binutils мы можем использовать аргумент *--with-lib-path* чтобы сконфигурировать пути поиска для библиотеки **ld**.

Для второго прохода сборки GCC, понадобится модификация исходных кодов этого пакета чтобы сообщить GCC использовать новый, созданный ранее динамический компоновщик. Несоблюдение этого требования приведет к тому, что программы будут использовать динамический компоновщик хост системы, который находится в каталоге `/lib` , что

противоречит задачи изоляции временного набора инструментов. Начиная с этого момента ядро временного набора инструментов полностью самодостаточна и независима. Остальная часть пакетов в главе Глава 5 будет скомпилирована с новым пакетом Glibc в каталоге `/tools` ..

При входе в среду chroot в главе Chapter 6, первый основной пакет, который будет скомпилирован - это пакет Glibc, в силу упомянутых выше аспектов самодостаточности. Как только Glibc будет установлен в каталог `/usr` мы выполним переключение с настроек по умолчанию временного набора инструментов и приступим к сборке остальной части.

5.3. Общие инструкции по компиляции

При создании пакетов существует несколько допущений, сделанных в рамках инструкции:

- К некоторым пакетам необходимо применить патч, перед началом процесса компиляции, но только тогда, когда применение патча необходимо чтобы обойти проблему. Применение патча обычно требуется в этой главе и следующей, а иногда только в этой или в следующей. Поэтому не нужно беспокоится если инструкции по загрузке патча будут отсутствовать. Предупреждения такие как *offset* или *fuzz* могут также возникать при применении патча. Не беспокойтесь по поводу этих предупреждений, если патч применен успешно.
- Во время процесса компиляции многих пакетов, на экране будут отображаться различные предупреждения. Это нормально, и можно об этом не беспокоится. Многие предупреждения будут указывать о том, что используются устаревший или не валидный синтаксис языка C или C++. Стандарты языка C выходят достаточно часто, и некоторые пакеты по прежнему могут использовать устаревший стандарт. Это не является проблемой, но может вызвать предупреждение.
- Еще раз выполните проверку, и убедитесь, что переменная окружения LFS установлена правильно:

```
echo $LFS
```

  Убедитесь что вывод указывает на путь к точке монтирования раздела LFS, который находится в каталоге `/mnt/lfs` , как предлагается в этой книге для примера.

- Наконец, необходимо подчеркнуть два важных момента:

Important

В инструкциях по сборке предполагается, что в главе Host System Requirements, а также что все необходимые символические ссылки ,указанны правильно:

- **bash** используемая оболочка
- **sh** символическая ссылка на **bash**.
- **/usr/bin/awk** символическая ссылка на **gawk**.
- **/usr/bin/yacc** символическая ссылка на **bison** или файл сценария, которая будет выполняться программой bison.

Important

Чтобы выполнить процесс сборки

1. Сохраните все файлы исходных кодов пакетов и патчи в каталог, который будет доступен из среды chroot, например `/mnt/lfs/sources/` . Не следует сохранять исходные коды пакетов и патчи в каталог `/mnt/lfs/tools/` .
2. Перейдите в каталог с исходными кодами
3. Для каждого пакета:
   a. Используйте команду **tar** для распаковки пакета, который необходимо скомпилировать. В главе 5, убедитесь что вы выполняете команды пользователем *lfs* когда выполняете распаковку пакета.
   b. Перейдите в каталог, который был создан в процессе распаковки пакета
   c. Следуйте инструкциям книги чтобы выполнить сборку пакета.
   d. Вернитесь обратно в каталог с исходными кодами и патчами.
   e. Удалите каталог созданный в процессе распаковки пакета, если инструкции в книге не указывают на другие действия.

5.4. Binutils-2.32 - Проход 1

Пакет содержит компоновщик, ассемблер, и другие утилиты и инструменты для работы с объектными файлам. Программы в этом пакете необходимы для компиляции как большинства пакетов системы LFS, так и многих пакетов за её пределами.

Приблизительное время сборки: 1 SBU

Требуемое дисковое пространство: 580 MB

5.4.1. Установка кросс-пакета Binutils

Note

Вернитесь назад и перечитайте информацию в предыдущем разделе. Понимание значительно сэкономит вам время и поможет избежать дальнейших проблем.

Очень важно чтобы пакет BinUtils был скомпилирован первым. потому что пакеты Glibc и GCC в процессе конфигурирования выполняют проверки доступности компоновщика и ассемблера, чтобы определить какие функциональные возможности включить или выключить.

В документации пакета Binutils рекомендуется выполнять сборку в отдельном каталоге:

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

Note

Для того, чтобы воспользоваться значением SBU (Стандартная единица времени сборки) в процессе компиляции пакетов во всей книге, необходимо измерить время конфигурирования и последующей компиляции этого пакета. Это можно легко сделать. Необходимо обернуть команды следующей инструкцией:**`time { ./configure ... && ... && make install; }`**.

Note

Приблизительное время сборки, значение SBU и требуемое дисковое пространство в главе 5 не включает информацию по выполнению наборов тестов пакетов.

Необходимо подготовить пакет Binutils к компиляции:

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

Значение параметров конфигурации:

`--prefix=/tools`
Указывает сценарию configure подготовить установку программ пакета Binutils в каталог `/tools` .

`--with-sysroot=$LFS`
Для выполнения кросс-компиляции, значение передается в систему сборки для поиска в каталоге $LFS целевых системных библиотек по мере необходимости.

`--with-lib-path=/tools/lib`
Указывает в каком каталоге должен находится компоновщик.

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

`--disable-nls`
Отключает интернационализацию, поскольку i18n не требуется во временном наборе инструментов.

`--disable-werror`
Это предотвращает остановку сборки в том случае, если появляются предупреждения компилятора хост-системы.

Чтобы скомпилировать пакет, необходимо выполнить команду:

```
make
```

Компиляция завершена. Как правило, на данном этапе необходимо выполнить наборы тестов для пакета, но сейчас пока рано, потому что инструменты для выполнения тестирования (Tcl, Expect, and DejaGNU) пока не на своём месте. Преимущества использования тестов на данный момент минимальны, поскольку программы из сборки первого прохода будут заменены на программы из сборки второго прохода.

Если сборка происходит на 64-разрядной машине, необходимо создать символическую ссылку для нормальной работы временного набора инструментов.

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

Установите пакет:

```
make install
```

Подробная информация об этом пакете находится в Section 6.16.2, “Содержимое пакета Binutils.”

5.5. GCC-9.2.0 - Проход 1

Пакет содержит набор компиляторов GNU, для таких языков как Си и Си++.

Приблизительное время сборки: 12 SBU
Требуемое дисковое пространство: 3.1 GB

5.5.1. Установка кросс-пакета GCC

Пакет GCC требует пакеты GMP, MPFR и MPC. Так как эти пакеты могут не быть установлены в хост-системе, они будут скомпилированы вместе с GCC. Распакуйте каждый пакет в каталог с исходным кодом пакета GCC и переименуйте их чтобы эти пакеты использовались в процессе сборки GCC и были скомпилированы автоматически.

Note

В этом разделе часто возникают недоразумения. Процедуры таки же же как и в любом другом разделе, и полностью соответствуют инструкции по сборке пакетов (Инструкции по сборке пакетов). Сначала распакуйте пакет GCC из архива, который находится в каталоге с другими архивами пакетов и патчами, затем перейдите в распакованный каталог. Только после этого вы должны следовать приведенным ниже инструкциям.

```
tar -xf ../mpfr-4.0.2.tar.xz
mv -v mpfr-4.0.2 mpfr
tar -xf ../gmp-6.1.2.tar.xz
mv -v gmp-6.1.2 gmp
tar -xf ../mpc-1.1.0.tar.gz
mv -v mpc-1.1.0 mpc
```

Следующая команда изменит место расположения динамического компоновщика чтобы использовать тот, который находится в каталоге `/tools` . Команда также удалит каталог `/usr/include` из путей поиска GCC. Выполните команду:

```
for file in gcc/config/{linux,i386/linux{,64}}.h
do
  cp -uv $file{,.orig}
  sed -e 's@/lib\(64\)\?\(32\)\?/ld@/tools&@g' \
      -e 's@/usr@/tools@g' $file.orig > $file
  echo '
#undef STANDARD_STARTFILE_PREFIX_1
#undef STANDARD_STARTFILE_PREFIX_2
#define STANDARD_STARTFILE_PREFIX_1 "/tools/lib/"
#define STANDARD_STARTFILE_PREFIX_2 ""' >> $file
  touch $file.orig
done
```

Если команда приведенная выше кажется непонятной, давайте разберем ее содержимое. Сначала мы копируем файлы `gcc/config/linux.h` , `gcc/config/i386/linux.h` ,и `gcc/config/i368/linux64.h` в файлы с таким же наименованием, но с добавлением суффикса

".orig". Затем первое выражение добавляет "/tools" к каждому экземпляру "/lib/ld", "/lib64/ld" или "/lib32/ld" в то время как второй заменяет жестко закодированные экземпляры "/usr". Потом мы добавляем нашу директиву define которая изменяет префикс начальных файлов на префикс конечных файлов. Обратите внимание что начальный символ "/" в "/tools/lib/" является обязательным. Потом мы используем команду **touch** для обновления даты и времени модификации скопированных файлов. При использовании в сочетании с **cp -u** предотвращает непредвиденные изменения исходных файлов, если команды случайно выполнили дважды.

Наконец, для хост-системы x86_64 устанавливаем каталогом по умолчанию наименование 64-битных библиотек в "lib":

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

В документации пакета GCC рекомендуется выполнять сборку в отдельном каталоге:

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

Подготовьте пакет GCC к компиляции:

```
../configure                                       \
    --target=$LFS_TGT                              \
    --prefix=/tools                                \
    --with-glibc-version=2.11                      \
    --with-sysroot=$LFS                            \
    --with-newlib                                  \
    --without-headers                              \
    --with-local-prefix=/tools                     \
    --with-native-system-header-dir=/tools/include \
    --disable-nls                                  \
    --disable-shared                               \
    --disable-multilib                             \
    --disable-decimal-float                        \
    --disable-threads                              \
    --disable-libatomic                            \
    --disable-libgomp                              \
    --disable-libquadmath                          \
    --disable-libssp                               \
    --disable-libvtv                               \
    --disable-libstdcxx                            \
    --enable-languages=c,c++
```

Значение параметров конфигурации:

`--with-newlib`

Так как рабочая библиотека C пока недоступна, необходимо убедиться в том, что константа inhibit_libc определена, когда выполняется сборка libgcc. Также это предотвратит компиляцию любого кода, которому необходима поддержка библиотеки libc.

`--without-headers`

Когда создается кросс-компилятор, пакету GCC необходимы стандартные заголовки, совместимые с целевой системой. Для требуемой задачи эти заголовочные файлы не нужны. Этот аргумент предотвращает поиск этих файлов пакетом GCC.

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

Этот аргумент (локальный префикс) является местом в системе, где GCC будет искать локально установленные include файлы. По умолчанию этот путь `/usr/local` . Указание на каталог `/tools` поможет оставить каталог `/usr/local` хост-системы за пределами поиска файлов устанавливаемого пакета GCC.

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

По умолчанию, GCC выполняет поиск системных заголовочных файлов в каталоге `/usr/include` . В сочетании с аргументом sysroot, поиск системных заголовочных файлов будет производится в каталоге `$LFS/tools/include` .Однако заголовочные файлы , которые будут установлены в следующих двух разделах, будут располагаться в каталоге `$LFS/tools/include` . Этот аргумент позволит искать их по указанному в значении пути. Вторым проходом сборки GCC, этот же аргумент обеспечит отсутствие заголовочных файлов из хост-системы.

`--disable-shared`

Аргумент указывает GCC, что необходимо произвести статическую линковку внутренних библиотек. Ето необходимо сделать, чтобы избежать возможных проблем с хост-системой.

`--disable-decimal-float`, *`--disable-threads`*, *`--disable-libatomic`*, *`--disable-libgomp`*, *`--disable-libquadmath`*, *`--disable-libssp`*, *`--disable-libvtv`*, *`--disable-libstdcxx`*

Этот аргумент отключает поддержку расширений для работы с десятичными числами и числами с плавающей запятой, потоками, библиотеками libatomic, libgomp, libquadmath, libssp, libvtv и стандартной библиотеки C++ соответственно. При компиляции этих функций, возникнет ошибка и для кросс-компилятора, эти функции не нужны, для того чтобы выполнить кросс-компиляцию временной библиотеки C (libc).

`--disable-multilib`

Для платформы x86_64, LFS пока не поддерживает конфигурацию multilib. Этот аргумент ни как не повлияет, если установка выполняется на платформе x86.

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

Этот параметр гарантирует, что будут созданы только компиляторы C и C ++. Это единственные языки, которые нужны сейчас.

Чтобы скомпилировать пакет, необходимо выполнить команду:

```
make
```

Компиляция завершена. Как правило, на данном этапе необходимо выполнить наборы тестов для пакета, но сейчас пока рано, потому что инструменты для выполнения тестирования (Tcl, Expect, and DejaGNU) пока не на своём месте. Преимущества использования тестов на данный момент минимальны, поскольку программы из сборки первого прохода будут заменены на программы из сборки второго прохода.

Установите пакет:

```
make install
```

Подробная информация об этом пакете находится в Section 6.21.2, “Содержимое пакета GCC.”

5.6. Заголовочные файлы Linux-5.2.8

Заголовочные файлы Linux API (в linux-5.2.8.tar.xz) предоставляют API для использования его библиотекой C (Glibc).

Приблизительное время сборки: 0.1 SBU
Требуемое дисковое пространство: 960 MB

5.6.1. Установка заголовочных файлов

Ядро Linux должно предоставлять интерфейс (API) для его использования системной библиотекой C (Glibc в LFS).Это можно сделать, выделив различные заголовочные файлы C которые расположены в архиве исходных кодов пакета ядра.

Необходимо убедиться, что в пакете нет устаревших файлов:

```
make mrproper
```

Теперь установите видимые пользователям заголовочные файлы из каталога исходных кодов пакета ядра. Они будут расположены в промежуточном каталоге и будут скопированы в указанное местоположение , потому что процесс извлечения удаляет все существующие файлы в целевом каталоге.

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

Подробная информация об этом пакете находится в Section 6.7.2, “Содержимое пакета Linux API Headers.”

5.7. Glibc-2.30

Пакет содержит стандартную библиотеку языка Си (GNU C Library). Эта библиотека предоставляет функции для выделения памяти, поиска каталогов, открытия и закрытия файлов, чтения и записи файлов, обработку строк, соответствия шаблонов (pattern matching), арифметические операции, и так далее.

Приблизительное время сборки: 4.8 SBU
Требуемое дисковое пространство: 896 MB

5.7.1. Установка пакета Glibc

В документации пакета Glibc рекомендуется выполнять сборку в отдельном каталоге:

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

Подготовьте пакет Glibc к компиляции:

```
../configure                             \
      --prefix=/tools                    \
      --host=$LFS_TGT                    \
      --build=$(../scripts/config.guess) \
      --enable-kernel=3.2              \
      --with-headers=/tools/include
```

Значение параметров конфигурации:

`--host=$LFS_TGT, --build=$(../scripts/config.guess)`
Комбинация этих аргументов указывает системе сборки Glibc выполнять кросс-компиляцию пакета, используя кросс-компоновщик и кросс-компилятор из каталога `/tools` .

`--enable-kernel=3.2`
Аргумент позволяет выполнять компиляцию с поддержкой ядер начиная с версии 3.2 и более поздних. Поддержка более старых ядер не будет включена.

`--with-headers=/tools/include`
Аргумент позволяет выполнять компиляцию с указанием заголовочных файлов, установленных в каталог с временным набором инструментов, таким образом пакету будет известно, какие функции ядра используются, и будет выполнена оптимизация.

На этапе конфигурирования могут появиться следующие предупреждения:

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

Отсутствие или несовместимость программы **msgfmt** безвредно. Это программа является частью пакета Gettext, который должен быть предоставлен хост-системой.

Note

Сообщалось, что этот пакет может не компилироваться при выполнении процедуры "параллельной сборки". Если это произойдет, повторите команду make с аргументом "-j1".

Скомпилируйте пакет:

```
make
```

Установите пакет:

```
make install
```

Caution

На этом этапе, необходимо остановиться, и проверить что базовый функционал (компиляция и линовка) созданных инструментов работает так, как необходимо. Чтобы выполнить проверку работоспособности, выполните следующие команды:

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

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

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

Обратите внимание, что на 32-битной машине наименование интерпретатора будет: `/tools/lib/ld-linux.so.2` .

Если вывод не соответствует тому, как показано выше, или вообще вывода нет, значит,что-то не так. Изучите и повторите шаги выше, чтобы выяснить где проблема и устраните её. Важно, устранить проблему до того, как вы продолжите следующие шаги.

Когда все будет хорошо, очистите тестовые файлы:

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

Note

При сборке пакета Binutils в следующем разделе, будут производиться дополнительные проверки, что временный набор инструментов был собран правильно. Если пакет Binutils не получится скомпилировать, или сконфигурировать, это будет явный признак того, что что-то пошло не так с предыдущими установками пакетов Binutils, GCC, или Glibc.

Подробная информация об этом пакете находится в Section 6.9.3, “Содержимое пакета Glibc.”

5.8. Libstdc++ из пакета GCC-9.2.0

Libstdc++ является стандартной библиотекой языка Си++. Она необходима для правильной работы компилятора g++ (часть GCC написана на языке Си++). Сборку пришлось отложить на этапе gcc-pass1 потому что присутствует зависимость с glibc, которая еще недоступна в каталоге /tools.

Приблизительное время сборки: 0.5 SBU
Требуемое дисковое пространство: 879 MB

5.8.1. Установка целевой библиотеки Libstdc++

Note

Libstdc++ является частью исходных кодов пакета GCC. Сначала необходимо распаковать архив с исходным кодом пакета GCC и перейти в каталог `gcc-9.2.0` .

Создайте отдельный каталог для сборки Libstdc++:

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

Подготовьте пакет Libstdc++ к компиляции:

```
../libstdc++-v3/configure           \
    --host=$LFS_TGT                 \
    --prefix=/tools                 \
    --disable-multilib              \
    --disable-nls                   \
    --disable-libstdcxx-threads     \
    --disable-libstdcxx-pch         \
    --with-gxx-include-dir=/tools/$LFS_TGT/include/c++/9.2.0
```

Значение параметров конфигурации:

`--host=...`
Указывает на использование кросс-компилятора который был установлен вместо компилятора в каталоге `/usr/bin` .

`--disable-libstdcxx-threads`
Поскольку сборка библиотеки Си для работы с потоками еще не создана, соответствующую библиотеку Си++ скомпилировать тоже не получится.

`--disable-libstdcxx-pch`
Этот аргумент указывает на то, что не требуется установка предварительно скомпилированых заголовочных файлов, которые на данном этапе не требуются.

`--with-gxx-include-dir=/tools/$LFS_TGT/include/c++/9.2.0`
Это каталог, где будет выполнятся поиск стандартных заголовочных файлов компилятором Си++. При обычной сборке, эта информация автоматически передается сценарию **configure**

пакета Libstdc++ из каталога верхнего уровня. В нашем случае, эта информация должны быть указана явно.

Чтобы скомпилировать пакет Libstdc++, необходимо выполнить команду:

```
make
```

Установите бибоиотеку:

```
make install
```

Подробная информация об этом пакете находится в Section 6.21.2, “Содержимое пакета GCC.”

5.9. Binutils-2.32 - Проход 2

Пакет содержит компоновщик, ассемблер, и другие утилиты и инструменты для работы с объектными файлам. Программы в этом пакете необходимы для компиляции как большинства пакетов системы LFS, так и многих пакетов за её пределами.

Приблизительное время сборки: 1.1 SBU
Требуемое дисковое пространство: 879 MB

5.9.1. Установка пакета Binutils

Необходимо снова создать отдельный каталог:

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

Подготовьте пакет Binutils к компиляции:

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

Значения новых параметров конфигурации:

`CC=$LFS_TGT-gcc AR=$LFS_TGT-ar RANLIB=$LFS_TGT-ranlib`
Поскольку это нативная сборка пакета Binutils, указание этих переменных гарантирует что система сборки пакета будет использовать кросс-компилятор и связанные с ним инструменты, вместо инструментов хост-системы.

`--with-lib-path=/tools/lib`
Указание аргумента сообщает сценарию configure использовать каталог для поиска библиотек, указанный в значении аргумента во время процесса компиляции пакета Binutils, в результате чего каталог `/tools/lib` будет передан компоновщику. Это предотвратит использование компоновщика хост-системы.

`--with-sysroot`
Функция sysroot позволяет компоновщику находить общие объекты которые требуются для других общих объектов, явно включенных в команду компоновщика. Без указания этого аргумента, на некоторых хост-системах, компиляция пакетов может не выполнится.

Скомпилируйте пакет:

```
make
```

Установите пакет:

```
make install
```

Теперь, подготовьте компоновщик, для “перенастройки” в следующей главе:

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

Значение параметров команды make:

`-C ld clean`
: Аргумент сообщает программе make удалить все скомпилированые файлы в подкаталоге `ld`.

`-C ld LIB_PATH=/usr/lib:/lib`
: Этот аргумент сообщает выполнить пересборку всего содержимого в подкаталоге `ld`. Указание переменной `LIB_PATH` Makefile в командной строке позволяет переопределить значения по умолчанию для набора временных инструментов указанием правильного конечного пути. Значение этой переменной указывает компоновщику каталог для поиска библиотек. Эта подготовка требуется для выполнения процедур в следующем разделе.

Подробная информация об этом пакете находится в Section 6.16.2, “Содержимое пакета Binutils.”

5.10. GCC-9.2.0 - Проход 2

Пакет содержит набор компиляторов GNU, для таких языков как Си и Си++.

Приблизительное время сборки: 15 SBU
Требуемое дисковое пространство: 3.7 GB

5.10.1. Установка пакета GCC

В первом проходе сборки GCC были установлены некоторые внутренние заголовочные файлы. Обычно некоторые из них, например - `limits.h` , будет включен в систему как заголовочный файл `limits.h` , в нашем случае в каталог `/tools/include/limits.h` . Однако в первом проходе сборки файл `/tools/include/limits.h` не существует, таким образом внутренний заголовочный файл установленный GCC является частичным, автономный файл и не содержит функции системного заголовочного файла. Этого достаточно, для выполнения сборки временной библиотеки libc, но теперь, пакету GCC необходимы все внутренние заголовочные файлы. Создание полной версии внутренних заголовочных файлов используя идентичную команду, при которой система сборки GCC работает при обычных обстоятельствах:

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

Снова, изменим местоположение динамического компоновщика пакета GCC по умолчанию используя тот, который установлен в каталоге `/tools` .

```
for file in gcc/config/{linux,i386/linux{,64}}.h
do
  cp -uv $file{,.orig}
  sed -e 's@/lib\(64\)\?\(32\)\?/ld@/tools&@g' \
      -e 's@/usr@/tools@g' $file.orig > $file
  echo '
#undef STANDARD_STARTFILE_PREFIX_1
#undef STANDARD_STARTFILE_PREFIX_2
#define STANDARD_STARTFILE_PREFIX_1 "/tools/lib/"
#define STANDARD_STARTFILE_PREFIX_2 ""' >> $file
  touch $file.orig
done
```

Для хост-системы x86_64 устанавливаем каталогом по умолчанию наименование 64-битных библиотек в “lib”:

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

Также, как и в первом проходе сборки GCC необходимы пакеты GMP, MPFR и MPC. распакуйте архивы с исходными кодами этих пакетов и переместите их требуемые каталоги:

```
tar -xf ../mpfr-4.0.2.tar.xz
mv -v mpfr-4.0.2 mpfr
tar -xf ../gmp-6.1.2.tar.xz
mv -v gmp-6.1.2 gmp
tar -xf ../mpc-1.1.0.tar.gz
mv -v mpc-1.1.0 mpc
```

Создайте снова отдельный каталог для сборки:

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

Перед началом сборки GCC, не забудьте очистить все переменные окружения, которые переопределяют аргументы и флаги для оптимизации.

Подготовьте пакет GCC к компиляции:

```
CC=$LFS_TGT-gcc                                         \
CXX=$LFS_TGT-g++                                        \
AR=$LFS_TGT-ar                                          \
RANLIB=$LFS_TGT-ranlib                                  \
../configure                                            \
    --prefix=/tools                                     \
    --with-local-prefix=/tools                          \
    --with-native-system-header-dir=/tools/include      \
    --enable-languages=c,c++                            \
    --disable-libstdcxx-pch                             \
    --disable-multilib                                  \
    --disable-bootstrap                                 \
    --disable-libgomp
```

Значения новых параметров конфигурации:

`--enable-languages=c,c++`
Этот аргумент гарантирует, что будут собраны оба компилятора C и C ++.

`--disable-libstdcxx-pch`
Не выполнят сборку предварительно скомпилированых заголовочных файлов (PCH) для `libstdc++`. Они занимают много места, и их использование не предусмотрено.

`--disable-bootstrap`
Для нативной сборки пакета GCC по умолчанию используется механизм сборки bootstrap. Он не просто компилирует GCC, а выполняет компиляцию в несколько проходов. Он использует программы, скомпилированые в первом проходе, чтобы выполнить повторную компиляцию на втором проходе, и затем снова в третий раз. Результат компиляции второго и третьего прохода сравниваются чтобы убедиться в безотказности. Также выполняется проверка на предмет корректности компиляции. Однако метод сборки LFS должен обеспечивать надежный компилятор без использования механизма сборки bootstrap каждый раз.

Скомпилируйте пакет:

```
make
```

Установите пакет:

```
make install
```

После установки, необходимо создать символические ссылки. Многие программы и файлы сценариев запускают **cc** вместо **gcc**, для обеспечения переносимости на все UNIX системы.Не у всех установлен именно компилятор GNU C. Запуск cc позволяет администратору выбирать, какой компилятор C устанавливать в систему, и мы создаем ссылку на него:

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

Caution

На этом этапе, необходимо остановиться, и проверить что базовый функционал (компиляция и линовка) созданных инструментов работает так, как необходимо. Чтобы выполнить проверку работоспособности, выполните следующие команды:

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

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

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

Обратите внимание, что для 32-битной машины, каталог динамического компоновщика будет `/tools/lib/ld-linux.so.2` .

Если вывод не соответствует тому, как показано выше, или вообще вывода нет, значит,что-то не так. Изучите и повторите шаги выше, чтобы выяснить где проблема и устраните её. Важно, устранить проблему до того, как вы продолжите следующие шаги. Во-первых, повторите проверку работоспособности **gcc** вместо **cc**. Если все работает, тогда символическая ссылка `/tools/bin/cc` отсутствует. Далее, убедитесь что переменная окружения PATH указана корректно. Это можно проверить, запустив команду **echo $PATH** и проверить что каталог `/tools/bin` находиться в начале списка. Если переменная окружения некорректная, это может означать, что вы авторизованы не как пользователь `lfs` или что то пошло не так на более ранних этапах. Прочитайте внимательно главу Section 4.4, “Настройка окружения.” ещё раз.

Если всё в порядке, удалите тестовые файлы:

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

Подробная информация об этом пакете находится в Section 6.21.2, “Содержимое пакета GCC.”

5.11. Tcl-8.6.9

Пакет содержит "командный язык инструментов" - скриптовый язык высокого уровня. Он необходим для выполнения тестов некоторых пакетов LFS, и будет установлен только во временный инструментарий.

Приблизительное время сборки: 0.9 SBU
Требуемое дисковое пространство: 71 MB

5.11.1. Установка пакета Tcl

Этот пакет, и следующие два (Expect и DejaGNU) будут установлены для возможности запуска набора тестов для пакетов GCC и Binutils и других. Установка трех пакетов для тестирования может показаться чрезмерным, но очень обнадеживает, если важно знать что скомпилированые инструменты работают правильно. Выполнение тестов в этой главе не является обязательным, эти пакеты необходимы для выполнения набора тестов в главе Chapter 6.

Обратите внимание, что используется минимальная версия пакета Tcl для запуска тестов LFS. Полный пакет доступ в *BLFS* .

Подготовьте пакет Tcl к компиляции:

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

Выполните сборку пакета:

```
make
```

Компиляция завершена. Как обсуждалось ранее выполнение тестов не является обязательным для набора временных инструментов в этой главе. Чтобы запустить выполнение тестов, в любом случае, выполните следующую команду:

```
TZ=UTC make test
```

Наборы тестов Tcl могут завершаться с ошибками при определённых условиях в хост-системе, которые до конца не понятны. Следовательно, сбои при выполнении тестов набора вполне ожидаемы, и они не считаются критичными. Установка аргумента *TZ=UTC* устанавливает часовой пояс для Всемирного координированного времени (UTC) только на время выполнения тестового набора. Это гарантирует, что тесты времени осуществляются правильно. Информацию о переменной окружения TZ можно посмотреть в главе Chapter 7.

Установите пакет:

```
make install
```

Сделайте установленную библиотеку доступной для записи, чтобы отладочные символы могли быть удалены позднее:

```
chmod -v u+w /tools/lib/libtcl8.6.so
```

Установите заголовочные файлы Tcl. Следующему пакету, Expect, они необходимы чтобы выполнить сборку.

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

Создайте необходимую символическую ссылку:

```
ln -sv tclsh8.6 /tools/bin/tclsh
```

5.11.2. Содержимое пакета Tcl

Установленные программы:	tclsh (link to tclsh8.6) and tclsh8.6
Установленная библиотека:	libtcl8.6.so, libtclstub8.6.a

Краткое описание

tclsh8.6	Командная оболочка Tcl
tclsh	Ссылка на tclsh8.6
`libtcl8.6.so`	Библиотека Tcl
`libtclstub8.6.a`	Библиотека Tcl Stub

5.12. Expect-5.45.4

Пакет содержит инструменты для автоматизации и тестирования, и является расширением к скрипт-языку Tcl, для многих интерактивных приложений. Он будет установлен только во временный инструментарий.

Приблизительное время сборки: 0.1 SBU
Требуемое дисковое пространство: 4.0 MB

5.12.1. Установка пакета Expect

Для начала, необходимо указать сценарию конфигурирования пакета использовать `/bin/stty` вместо `/usr/local/bin/stty` который он может искать в хост-системе. Это гарантирует что наши инструменты тестирования останутся в нормальном состоянии до окончания сборки всего временного набора инструментов:

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

Подготовьте пакет Expect к компиляции:

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

Значение параметров конфигурации:

`--with-tcl=/tools/lib`
Аргумент обеспечивает то что сценарий конфигурирования будет выполнять поиск установленного пакета Tcl во каталогах временного инструментария, вместо поиска в каталогах хост-системы.

`--with-tclinclude=/tools/include`
Значение аргумента явно указывает, где выполнять поиск внутренних заголовочных файлов пакета Tcl. Использование этой опции позволяет избежать условий, при которых сценарий **configure** может не выполнится, потому что не сможет автоматически найти заголовочные файлы Tcl.

Выполните сборку пакета:

```
make
```

Компиляция завершена. Как обсуждалось ранее выполнение тестов не является обязательным для набора временных инструментов в этой главе. Чтобы запустить выполнение тестов, в любом случае, выполните следующую команду:

```
make test
```

Наборы тестов Tcl могут завершаться с ошибками, которые не возможно контролировать. Следовательно, сбои при выполнении тестов набора вполне ожидаемы, и они не считаются критичными.

Установите пакет:

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

Значение параметров make:

SCRIPTS=""
Это предотвращает установку дополнительных сценариев Expect, которые не нужны.

5.12.2. Содержимое пакета Expect

Установленная программа: expect
Установленная библиотека: libexpect-5.45.so

Краткое описание

expect Общается с другими интерактивными программами в соответствии со сценарием

`libexpect-5.45.so` Содержит функции, которые позволяют пакету Expect быть использованным как расширение к Tcl или использоваться напрямую, из Си или Си++ (без Tcl)

5.13. DejaGNU-1.6.2

The DejaGNU пакет содержит a framework for testing other programs.

Приблизительное время сборки: less than 0.1 SBU
Требуемое дисковое пространство: 3.2 MB

5.13.1. Установка пакета DejaGNU

Выполните подготовку пакета DejaGNU к компиляции:

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

Выполните сборку и установку пакета:

```
make install
```

Для выполнения тестов, выполните команду:

```
make check
```

5.13.2. Содержимое пакета DejaGNU

Установленная программа: runtest

Краткое описание

runtest Сценарий-обёртка, который находит правильную оболочку **expect** и затем, выполняет запуск DejaGNU

5.14. M4-1.4.18

Пакет содержит общий макропроцессор текста - полезный инструмент для выполнения сборки других программ.

Приблизительное время сборки: 0.2 SBU
Требуемое дисковое пространство: 20 MB

5.14.1. Установка пакета M4

Сначала, сделаем некоторые исправления, требуемые glibc-2.28:

```
sed -i 's/IO_ftrylockfile/IO_EOF_SEEN/' lib/*.c
echo "#define _IO_IN_BACKUP 0x100" >> lib/stdio-impl.h
```

Выполните подготовку пакета M4 к компиляции:

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

Скомпилируйте пакет:

```
make
```

Компиляция завершена. Как обсуждалось ранее выполнение тестов не является обязательным для набора временных инструментов в этой главе. Чтобы запустить выполнение тестов, в любом случае, выполните следующую команду:

```
make check
```

Установите пакет:

```
make install
```

Подробная информация об этом пакете находится в Section 6.14.2, “Содержимое пакета M4.”

5.15. Ncurses-6.1

Пакет содержит библиотеку, предназначенную для управления вводом-выводом на терминал, в числе прочего, библиотека позволяет задавать экранные координаты (в знакоместах) и цвет выводимых символов. Предоставляет программисту уровень абстракции, позволяющий не беспокоиться об аппаратных различиях терминалов и писать переносимый код. Он необходим для ряда пакетов.

Приблизительное время сборки: 0.6 SBU
Требуемое дисковое пространство: 41 MB

5.15.1. Установка пакета Ncurses

Для начала, необходимо убедиться в том, что программа **gawk** обнаружится первой, в процессе конфигурирования:

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

Подготовьте пакет Ncurses к компиляции:

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

Значение параметров конфигурации:

`--without-ada`
Указание этого аргумента гарантирует что поддержка компилятора Ada не будет включена в сборку, который может присутствовать на хост-системе, но не будет доступен в **chroot** окружении.

`--enable-overwrite`
Указание этого аргумента сообщает что необходимо установить заголовочные файлы в каталог `/tools/include`, вместо `/tools/include/ncurses`, чтобы быть уверенным в том, что остальные пакеты смогут легко найти заголовочные файлы пакета Ncurses.

`--enable-widec`
Аргумент сообщает, что необходимо подключить wide-character библиотеки (например, `libncursesw.so.6.1`) вместо обычных библиотек (например, `libncurses.so.6.1`). Библиотеки wide-character могут использоваться как для многобайтовых, так и для традиционных 8-битных локалях, в то время как обычные библиотеки работают правильно только с 8-битными локалями. Библиотеки wide-character и обычные совместимы на уровне исходного кода, но не совместимы на бинарном уровне.

Скомпилируйте пакет:

```
make
```

В пакет включен набор тестов, но они могут быть запущены только после того, как пакет будет установлен. Тесты находятся в каталоге `test/`. Прочитайте файл `README` в этом каталоге для получения дополнительной информации.

Установите пакет:

```
make install
ln -s libncursesw.so /tools/lib/libncurses.so
```

Подробная информация об этом пакете находится в Section 6.24.2, “Содержимое пакета Ncurses.”

5.16. Bash-5.0

Усовершенствованная и модернизированная вариация командной оболочки Bourne shell. Этот пакет выполняет требования стандарта LFS Core для обеспечения интерфейса Bourne Shell в системе. Он был выбран из числа других оболочек из-за широкого распространения, возможностей которые выходят далеко за пределы базовых функций программ-оболочек.

Приблизительное время сборки: 0.4 SBU
Требуемое дисковое пространство: 67 MB

5.16.1. Установка пакета Bash

Подготовьте пакет Bash к компиляции:

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

Значение параметров конфигурации:

`--without-bash-malloc`
Эта опция отключает использование функций выделения памяти Bash (`malloc`) которые, как известно, вызывают ошибки сегментации. При использовании этого аргумента, Bash будет использовать более надежные функции `malloc` из библиотеки Glibc.

Скомпилируйте пакет:

```
make
```

Компиляция завершена. Как обсуждалось ранее выполнение тестов не является обязательным для набора временных инструментов в этой главе. Чтобы запустить выполнение тестов, в любом случае, выполните следующую команду:

```
make tests
```

Установите пакет:

```
make install
```

Создайте символическую ссылку для программ, которые используют **sh** для работы с оболочкой:

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

Подробная информация об этом пакете находится в Section 6.34.2, “Содержимое пакета Bash.”

5.17. Bison-3.4.1

Пакет содержит GNU версию yacc (Ещё один компилятор компиляторов) необходимый для сборки некоторых пакетов LFS системы.

Приблизительное время сборки: 0.3 SBU
Требуемое дисковое пространство: 39 MB

5.17.1. Установка пакета Bison

Подготовьте пакет Bison к компиляции:

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

Скомпилируйте пакет:

```
make
```

Для выполнения тестов, выполните команду:

```
make check
```

Установите пакет:

```
make install
```

Подробная информация об этом пакете находится в Section 6.31.2, “Содержимое пакета Bison.”

5.18. Bzip2-1.0.8

Пакет содержит программы для сжатия и распаковки файлов. Он необходим для распаковки многих пакетов LFS. Сжатие текстовых файлов при помощи программы **bzip2** даёт больший процент сжатия чем **gzip**.

Приблизительное время сборки: less than 0.1 SBU
Требуемое дисковое пространство: 6.0 MB

5.18.1. Установка пакета Bzip2

Пакет Bzip2 не содержит сценария конфигурации **configure**. Выполните компиляцию и тестирование, выполнив команду:

```
make
```

Установите пакет:

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

Подробная информация об этом пакете находится в Section 6.22.2, “Содержимое пакета Bzip2.”

5.19. Coreutils-8.31

Coreutils - пакет программного обеспечения GNU, содержащий большое количество основных утилит, таких как cat, ls и rm, необходимых для UNIX-подобных операционных систем. Пакет включает несколько более ранних пакетов — textutils, shellutils, fileutils и другие различные утилиты.

Приблизительное время сборки: 0.8 SBU
Требуемое дисковое пространство: 157 MB

5.19.1. Установка пакета Coreutils

Подготовьте пакет Coreutils к компиляции:

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

Значение параметров конфигурации:

`--enable-install-program=hostname`
Аргумент включает в установку программу **hostname**. Эта программа по умолчанию выключена, но она необходима для выполнения набора тестов пакета Perl.

Скомпилируйте пакет:

```
make
```

Компиляция завершена. Как обсуждалось ранее выполнение тестов не является обязательным для набора временных инструментов в этой главе. Чтобы запустить выполнение тестов, в любом случае, выполните следующую команду:

```
make RUN_EXPENSIVE_TESTS=yes check
```

The *RUN_EXPENSIVE_TESTS=yes* аргумент сообщает включить в набор ресурсозатратные (с точки зрения мощности процессора и использования памяти) тесты для некоторых платформ. Как правило, такие тесты не вызывают проблем в Linux.

Установите пакет:

```
make install
```

Подробная информация об этом пакете находится в Section 6.54.2, “Содержимое пакета Coreutils.”

5.20. Diffutils-3.7

Пакет содержит программы, которые отображают разницу в содержимом между файлами и каталогами. Эти программы могут быть использованы, для создания патчей, а также они используются в процедурах сборки для большинства пакетов.

Приблизительное время сборки: 0.2 SBU
Требуемое дисковое пространство: 26 MB

5.20.1. Установка пакета Diffutils

Подготовьте пакет Diffutils к компиляции:

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

Скомпилируйте пакет:

```
make
```

Компиляция завершена. Как обсуждалось ранее выполнение тестов не является обязательным для набора временных инструментов в этой главе. Чтобы запустить выполнение тестов, в любом случае, выполните следующую команду:

```
make check
```

Установите пакет:

```
make install
```

Подробная информация об этом пакете находится в Section 6.56.2, “Содержимое пакета Diffutils.”

5.21. File-5.37

Пакет содержит утилиты для определения типов файлов. Некоторым пакетам требуется, чтобы этот пакет был установлен.

Приблизительное время сборки: 0.1 SBU
Требуемое дисковое пространство: 19 MB

5.21.1. Установка пакета File

Подготовьте пакет File к компиляции:

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

Скомпилируйте пакет:

```
make
```

Компиляция завершена. Как обсуждалось ранее выполнение тестов не является обязательным для набора временных инструментов в этой главе. Чтобы запустить выполнение тестов, в любом случае, выполните следующую команду:

```
make check
```

Установите пакет:

```
make install
```

Подробная информация об этом пакете находится в Section 6.12.2, “Содержимое пакета File.”

5.22. Findutils-4.6.0

The Findutils пакет содержит программы для поиска файлов. Программы предоставляют способы для рекурсивного поиска файлов по дереву каталогов, создания, обслуживания и поиска в базе данных (как правильно поиск выполняется быстрее, чем рекурсивный поиск, но менее надёжен, если база данных не в актуальном состоянии).

Приблизительное время сборки: 0.3 SBU
Требуемое дисковое пространство: 36 MB

5.22.1. Установка пакета Findutils

Сначала, сделаем некоторые исправления, требуемые для glibc-2.28 и более поздних версий:

```
sed -i 's/IO_ftrylockfile/IO_EOF_SEEN/' gl/lib/*.c
sed -i '/unistd/a #include <sys/sysmacros.h>' gl/lib/mountlist.c
echo "#define _IO_IN_BACKUP 0x100" >> gl/lib/stdio-impl.h
```

Подготовьте пакет Findutils к компиляции:

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

Скомпилируйте пакет:

```
make
```

Компиляция завершена. Как обсуждалось ранее выполнение тестов не является обязательным для набора временных инструментов в этой главе. Чтобы запустить выполнение тестов, в любом случае, выполните следующую команду:

```
make check
```

Установите пакет:

```
make install
```

Подробная информация об этом пакете находится в Section 6.58.2, “Содержимое пакета Findutils.”

5.23. Gawk-5.0.1

Пакет содержит программы для манипуляции с текстовыми файлами. Это GNU версия awk (Aho-Weinberg-Kernighan). Он используется в процедурах сборки для большинства пакетов.

Приблизительное время сборки: 0.2 SBU
Требуемое дисковое пространство: 46 MB

5.23.1. Установка пакета Gawk

Подготовьте пакет Gawk к компиляции:

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

Скомпилируйте пакет:

```
make
```

Компиляция завершена. Как обсуждалось ранее выполнение тестов не является обязательным для набора временных инструментов в этой главе. Чтобы запустить выполнение тестов, в любом случае, выполните следующую команду:

```
make check
```

Установите пакет:

```
make install
```

Подробная информация об этом пакете находится в Section 6.57.2, “Содержимое пакета Gawk.”

5.24. Gettext-0.20.1

Пакет содержит утилиты и библиотеки для работы с локализацией и интернационализацией необходимые для некоторых пакетов. Что позволяет программам, которые скомпилированы с поддержкой NLS (Поддержка нативных языков), показывать сообщения в пользовательском нативном языке.

Приблизительное время сборки: 1.8 SBU
Требуемое дисковое пространство: 300 MB

5.24.1. Установка пакета Gettext

Для временного набора инструментов, необходимо скомпилировать и установить только три программы из пакета Gettext.

Подготовьте пакет Gettext к компиляции:

```
./configure --disable-shared
```

Значения параметров конфигурации:

`--disable-shared`
Не требуется установка и компиляция общих библиотек пакета Gettext.

Скомпилируйте пакет:

```
make
```

Due to the limited environment, running the test suite at this stage is not recommended.

Установите the **msgfmt**, **msgmerge** and **xgettext** programs:

```
cp -v gettext-tools/src/{msgfmt,msgmerge,xgettext} /tools/bin
```

Подробная информация об этом пакете находится в Section 6.47.2, “Содержимое пакета Gettext.”

5.25. Grep-3.3

Пакет содержит программу, которая находит на вводе строки, отвечающие заданному регулярному выражению, и выводит их, если вывод не отменён специальным ключом. Пакет используется в процедурах сборки для большинства пакетов.

Приблизительное время сборки: 0.2 SBU
Требуемое дисковое пространство: 24 MB

5.25.1. Установка пакета Grep

Подготовьте пакет Grep к компиляции:

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

Скомпилируйте пакет:

```
make
```

Компиляция завершена. Как обсуждалось ранее выполнение тестов не является обязательным для набора временных инструментов в этой главе. Чтобы запустить выполнение тестов, в любом случае, выполните следующую команду:

```
make check
```

Установите пакет:

```
make install
```

Подробная информация об этом пакете находится в Section 6.33.2, “Содержимое пакета Grep.”

5.26. Gzip-1.10

Пакет содержит программы для сжатия и распаковки файлов. Он необходим, чтобы выполнять распаковку большинства пакетов LFS.

Приблизительное время сборки: 0.1 SBU
Требуемое дисковое пространство: 10 MB

5.26.1. Установка пакета Gzip

Подготовьте пакет Gzip к компиляции:

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

Скомпилируйте пакет:

```
make
```

Компиляция завершена. Как обсуждалось ранее выполнение тестов не является обязательным для набора временных инструментов в этой главе. Чтобы запустить выполнение тестов, в любом случае, выполните следующую команду:

```
make check
```

Установите пакет:

```
make install
```

Подробная информация об этом пакете находится в Section 6.62.2, “Содержимое пакета Gzip.”

5.27. Make-4.2.1

Пакет содержит программы которые автоматизируют процесс преобразования файлов из одной формы в другую. Чаще всего это компиляция исходного кода в объектные файлы и последующая компоновка в исполняемые файлы или библиотеки. Он необходим для сборки пакетов LFS.

Приблизительное время сборки: 0.1 SBU
Требуемое дисковое пространство: 13 MB

5.27.1. Установка пакета Make

Для начала, необходимо обойти ошибку, из-за glibc-2.27 и более поздних версиях:

```
sed -i '211,217 d; 219,229 d; 232 d' glob/glob.c
```

Подготовьте пакет Make к компиляции:

```
./configure --prefix=/tools --without-guile
```

Значения параметров конфигурации:

`--without-guile`
Аргумент указывает программе Make-4.2.1 не линковаться с библиотеками Guile, которые могут быть на хост-системе, но отсутствовать в среде **chroot** в следующей главе.

Скомпилируйте пакет:

```
make
```

Компиляция завершена. Как обсуждалось ранее выполнение тестов не является обязательным для набора временных инструментов в этой главе. Чтобы запустить выполнение тестов, в любом случае, выполните следующую команду:

```
make check
```

Установите пакет:

```
make install
```

Подробная информация об этом пакете находится в Section 6.66.2, “Содержимое пакета Make.”

5.28. Patch-2.7.6

Пакет содержит программу предназначенную для переноса правок (изменений) между разными версиями текстовых файлов. Информация о правке обычно содержится в отдельном файле, называемом "заплаткой" (*patch*), "правкой" или "файлом правки" (англ. patch file). Подобный файл, как правило, создается с помощью другой утилиты Unix — diff, позволяющей автоматически извлечь информацию о различиях в тексте файлов. Он необходим для выполнения сборки некоторых пакетов LFS.

Приблизительное время сборки: 0.2 SBU
Требуемое дисковое пространство: 13 MB

5.28.1. Установка пакета Patch

Подготовьте пакет Patch к компиляции:

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

Скомпилируйте пакет:

```
make
```

Компиляция завершена. Как обсуждалось ранее выполнение тестов не является обязательным для набора временных инструментов в этой главе. Чтобы запустить выполнение тестов, в любом случае, выполните следующую команду:

```
make check
```

Установите пакет:

```
make install
```

Подробная информация об этом пакете находится в Section 6.67.2, “Содержимое пакета Patch.”

5.29. Perl-5.30.0

Высокоуровневый интерпретируемый динамический язык программирования общего назначения, он необходим для установки и выполнения тестов некоторых пакетов LFS.

Приблизительное время сборки: 1.6 SBU
Требуемое дисковое пространство: 275 MB

5.29.1. Установка пакета Perl

Подготовьте пакет Perl к компиляции::

```
sh Configure -des -Dprefix=/tools -Dlibs=-lm -Uloclibpth -Ulocincpth
```

Значения параметров конфигурации:

-des
: This is a combination of three options: -d uses defaults for all items; -e ensures completion of all tasks; -s silences non-essential output.

-Uloclibpth amd -Ulocincpth
: These entries undefine variables that cause the configuration to search for locally installed components that may exist on the host system.

Выполните сборку пакета:

```
make
```

Хотя Perl поставляется с набором тестов, было бы лучше подождать пока он не будет установлен в следующей главе.

На данный момент, только некоторые из утилит и библиотек должны быть установлены:

```
cp -v perl cpan/podlators/scripts/pod2man /tools/bin
mkdir -pv /tools/lib/perl5/5.30.0
cp -Rv lib/* /tools/lib/perl5/5.30.0
```

Подробная информация об этом пакете находится в Section 6.40.2, “Содержимое пакета Perl.”

5.30. Python-3.7.4

Python 3 - Высокоуровневый язык программирования общего назначения, ориентированный на повышение производительности разработчика и читаемости кода. Содержит среду разработки для объектно-ориентированного программирования, написания сценариев, прототипирования крупных программ и разработки полноценных приложений.

Приблизительное время сборки: 1.4 SBU
Требуемое дисковое пространство: 381 MB

5.30.1. Установка пакета Python

Note

Есть два файла пакета, имя которых начинается с “python”. Необхдимо распаковать пакет `Python-3.7.4.tar.xz` (первая буква - заглавная).

Пакет сначала выполняет сборку интерпретатора Python, затем, некоторые стандартные модули. Основной файл сценария для сборки модулей написан на Python, и использует жестко прописанные пути к каталогам хост системы `/usr/include` и `/usr/lib` . Необходимо предотвратить такое поведение и скорректировать файл сценария:

```
sed -i '/def add_multiarch_paths/a \        return' setup.py
```

Подготовьте пакет Python к компиляции::

```
./configure --prefix=/tools --without-ensurepip
```

Значение параметров конфигурации::

`--without-ensurepip`
Аргумент отключает установщик Python, котрый не требуется на этом этапе.

Скомпилируйте пакет:

```
make
```

Компиляция завершена. Для выполнения тестов требуется TK и X Windows, и сейчас, не могут быть запущены.

Установите пакет:

```
make install
```

Подробная информация об этом пакете находится в Section 6.51.2, “Содержимое пакета Python 3.”

5.31. Sed-4.7

Sed - потоковый текстовый редактор (а также язык программирования), применяющий различные предопределённые текстовые преобразования к последовательному потоку текстовых данных. Он необходим для многих пакетов LFS, на этапе конфигурирования.

Приблизительное время сборки: 0.2 SBU
Требуемое дисковое пространство: 20 MB

5.31.1. Установка пакета Sed

Подготовьте пакет Sed к компиляции:

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

Скомпилируйте пакет:

```
make
```

Компиляция завершена. Как обсуждалось ранее выполнение тестов не является обязательным для набора временных инструментов в этой главе. Чтобы запустить выполнение тестов, в любом случае, выполните следующую команду:

```
make check
```

Установите пакет:

```
make install
```

Подробная информация об этом пакете находится в Section 6.28.2, “Содержимое пакета Sed.”

5.32. Tar-1.32

Этот пакет обеспечивает возможности архивирования и извлечения почти всех пакетов, используемых в LFS.

Приблизительное время сборки: 0.3 SBU
Требуемое дисковое пространство: 38 MB

5.32.1. Установка пакета Tar

Подготовьте пакет Tar к компиляции:

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

Скомпилируйте пакет:

```
make
```

Компиляция завершена. Как обсуждалось ранее выполнение тестов не является обязательным для набора временных инструментов в этой главе. Чтобы запустить выполнение тестов, в любом случае, выполните следующую команду:

```
make check
```

Установите пакет:

```
make install
```

Подробная информация об этом пакете находится в Section 6.69.2, “Содержимое пакета Tar.”

5.33. Texinfo-6.6

Этот пакет - система документирования и язык разметки, позволяющие создавать документы в разных форматах из одного исходного текста. Он используется в процедурах установки многих пакетов LFS.

Приблизительное время сборки: 0.2 SBU
Требуемое дисковое пространство: 103 MB

5.33.1. Установка пакета Texinfo

Подготовьте пакет к компиляции:

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

Note

В процессе конфигурирования, могут быть отображены ошибки для TestXS_la-TestXS.lo..Эти ошибки не имеют никакого отношения к LFS, и их можно игнорировать.

Скомпилируйте пакет:

```
make
```

Компиляция завершена. Как обсуждалось ранее выполнение тестов не является обязательным для набора временных инструментов в этой главе. Чтобы запустить выполнение тестов, в любом случае, выполните следующую команду:

```
make check
```

Установите пакет:

```
make install
```

Подробная информация об этом пакете находится в Section 6.70.2, “Содержимое пакета Texinfo.”

5.34. Xz-5.2.4

Пакет содержит программы для сжатия и распаковки файлов. Он обеспечивает высокое сжатие и используется для распаковки пакетов форматов XZ и LZMA. Сжатие текстовых файлов при помощи программы **xz**даёт больший процент сжатия чем **gzip** или **bzip2**.

Приблизительное время сборки: 0.2 SBU
Требуемое дисковое пространство: 18 MB

5.34.1. Установка пакета Xz

Подготовьте пакет Xz к компиляции:

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

Скомпилируйте пакет:

```
make
```

Компиляция завершена. Как обсуждалось ранее выполнение тестов не является обязательным для набора временных инструментов в этой главе. Чтобы запустить выполнение тестов, в любом случае, выполните следующую команду:

```
make check
```

Установите пакет:

```
make install
```

Подробная информация об этом пакете находится в Section 6.45.2, “Содержимое пакета Xz.”

5.35. Удаление ненужных файлов

Инструкции в этом разделе являются необязательными для выполнения. Однако, если дисковый раздел небольшой, полезно знать какие элементы могут быть удалены. Исполняемые файлы и библиотеки, которые были скомпилированы, имеют файлы отладочных символов, общим размером около 70 Мегабайт. Их можно удалить, выполнив команду:

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

Этот набор команд пропустит некоторые файлы, сообщив что не удается распознать формат. Большинство из них являются файлами сценариев а не библиотеками. Также, используется системная команда очистки бинарных файлов в каталоге /tools.

Будьте осторожны, и *не* применяйте аргумент *--strip-unneeded* к библиотекам. Статические библиотеки будут уничтожены и пакеты временного набора инструментов придется компилировать заново.

Чтобы сохранить ещё больше места, можно удалить документацию:

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

Удаление ненужных файлов:

```
find /tools/{lib,libexec} -name \*.la -delete
```

На этом этапе у вас должно быть не менее 3 ГБ свободного места в каталоге $LFS, которое будет занято при сборке пакетов Glibc и Gcc на следующем этапе. Если можно выполнить сборку пакета Glibc, то можно создавать и устанавливать остальные пакеты также.

5.36. Изменение прав

Note

Команды в остальных частях книги должны выполняться от пользователя `root` и более не использовать временного пользователя `lfs`. Также, дважды проверьте что переменная окружения $LFS установлена пользователю `root`.

Сейчас, права на каталог `$LFS/tools` принадлежат пользователю `lfs`, который существует только в хост-системе. Если права на каталог оставить как есть, файлы будут иметь принадлежность к пользовательскому идентификатору но без соответствующей учетной записи. Это небезопасно, потому что в дальнейшем когда будут создаваться новые учетные записи, идентификатор может быть кому нибудь назначен, и права на этот каталог будут ему предоставлены, следовательно он сможет манипулировать всем содержимым каталога без каких либо ограничений.

Во избежание этой проблемы, позднее, можно добавить пользователя `lfs` в новую систему, когда будет создан файл `/etc/passwd` , позаботьтесь назначить ему тоже имя и идентификатор группы как в хост-системе. Лучше всего изменить права на каталог `$LFS/tools` , и назначить их пользователю `root`, запустив команду:

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

Несмотря на то, что каталог `$LFS/tools` может быть удалён, как только сборка системы LFS будет завершена, его можно сохранить для сборки другой системы LFS *той же версии книги.*. Каким образом выполнять резервное копирование каталога `$LFS/tools` является личным предпочтением.

Caution

Если вы намерены сохранить временный набор инструментов для повторной сборки системы LFS, *сейчас* самое время, чтобы выполнить резервное копирование. Команды, в процессе выполнения главы 6 изменять файлы и ссылки временного набора инструментов, и в конечном состоянии инструментарий будет бесполезным для будущих сборок.

Part III. Сборка системы LFS

Chapter 6. Установка основного системного программного обеспечения

6.1. Введение

В этой главе будет описан процесс сборки конечной системы LFS с помощью ранее установленного временного набора инструментов. С помощью команды chroot будет выполнен вход во временную мини Linux систему (временный набор инструментов), далее будут выполнены окончательные подготовительные работы, а затем будут выполнены установки необходимых пакетов.

Установка программного обеспечения проста. Несмотря на то, что инструкции по установке могли быть более компактны и универсальны, в книги предоставляются расширенные инструкции для каждого пакета, чтобы минимизировать возможные ошибки на этапе конфигурирования, сборки и тестирования. Ключом к пониманию что делает Linux систему рабочей, являются знания для чего используется каждый пакет и зачем он может понадобиться вам или системе.

Мы не рекомендуем использовать оптимизацию. Она может ускорить работу программы, но при этом могут возникнуть проблемы при компиляции и запуске. Если пакет не удается скомпилировать с включенной оптимизацией, необходимо её отключить, и выполнить повторную компиляцию. Даже если компиляция пакета была успешной, с включенной оптимизацией, существует риск что компиляция была выполнена неправильно, потому что могут быть сложные взаимодействия между кодом и инструментами сборки. Обратите внимание, что аргументы `-march` и `-mtune` не указаны в книге и не были протестированы. Они могут вызвать проблемы с программами набора временных инструментов, таких как Binutils, GCC и Glibc. Потенциальные преимущества при использовании оптимизации компилятора не так высоки по сравнению с рисками. Если сборка системы LFS выполняется Вами первый раз, рекомендуется не использовать оптимизацию. Система будет работать очень быстро и в тоже время будет стабильна.

Порядок установки пакетов в этой главе очень важен. Необходимо строго соблюдать последовательность сборки и установки пакетов, для гарантии того, что устанавливаемый пакет не будет иметь ссылку на каталог `/tools` . По этой же причине, не следует компилировать пакеты параллельно. Выполняя компиляцию параллельно, можно сэкономить время (особенно на многопроцессорных или многоядерных машинах), но это может привести к тому, что если программа имеет связи с каталогом `/tools` при выполнении она может перестать работать когда этот каталог будет удалён.

Перед инструкциями по установке, каждая страница содержит информацию о пакете, включая краткое описание содержимого пакета, приблизительное время сборки, и сколько дискового пространства требуется во время процесса сборки. После инструкций по установке, будет представлен список программ и библиотек (а также их краткое описание), которые включены в пакет.

Note

В Главе 6 значение SBU и требуемого дискового пространства содержит информацию по выполнению набора тестов для всех пакетов, где есть такая возможность.

6.1.1. О библиотеках

Как правило, редакторы LFS препятствуют созданию и установке статических библиотек. Первоначальная цель использования большинства статических библиотек в современных Linux системах устарела. Кроме того, связывание статической библиотеки в программу может быть не самым лучшим решением. Если требуется обновление библиотеки, для устранения проблем безопасности, то все программы, которые используют статическую библиотеку, должны выполнить повторную компоновку (линковку), чтобы изменить ссылку на новую версию библиотеки. Поскольку использование статических библиотек не всегда бывает очевидным, программы (и процедуры, необходимые для выполнения повторной компоновки) могут быть неизвестными.

В процедурах главы 6, будет удалена или закрыта возможность установки большинства статических библиотек. Как правило, такое правило передается аргументом `--disable-static` в сценарий **configure**. В некоторых случаях, необходимы альтернативные пути решения. Особенно это относится к пакетам glibc и gcc, где наличие статических библиотек остается неотъемлемым для общего процесса сборки пакетов.

Более подробное обсуждение вы можете изучить, перейдя по ссылке:*Библиотеки: Статические или динамические?* в книге BLFS.

6.2. Подготовка виртуальных файловых систем ядра

Различные файловые системы, экспортируемые ядром, используются для связи с ним. Они являются виртуальными, поскольку нет диска, на котором они бы использовались. Содержимое такие файловых систем располагается в памяти.

Выполните команду, чтобы создать структуру каталогов, куда будет выполнено монтирование виртуальных файловых систем.

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

6.2.1. Создание начальных узлов устройств

Когда ядро выполняет загрузку системы, ему требуется наличие нескольких устройств, таких как `console` и `null`. Эти узлы должны быть созданы на диске, для того, чтобы они были доступны после того, как запустится **udevd** и дополнительно, когда Linux запустится с *`init=/bin/bash`*. Для того, чтобы создать такие устройства, выполните команду:

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

6.2.2. Монтирование и заполнение каталога /dev

Рекомендуемый метод заполнения каталога (ссылки на устройства)`/dev` - примонтировать виртуальную файловую систему (такую, как `tmpfs`) в каталог `/dev`, и разрешить устройствам динамически создаваться на этих файловых системах по мере их обнаружения или возможности доступа к ним. Создание устройств обычно выполняется во время процесса загрузки программы Udev. Но поскольку в новой системе ещё отсутствует эта программа, следовательно устройства добавлены не будут. Необходимо вручную связать устройства из каталога `/dev` хост-системы.

Привязка - это особый вид монтирования, который позволяет создавать зеркало каталога в указанной точке монтирования в другом местоположении. Для того, чтобы это сделать, воспользуйтесь следующей командой:

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

6.2.3. Монтирование виртуальных файловых систем ядра

Теперь, необходимо примонтировать оставшиеся виртуальные файловые системы:

```
mount -vt devpts devpts $LFS/dev/pts -o gid=5,mode=620
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

Это гарантирует, что все созданные devpts узлы устройств будут принадлежать группе с идентификатором 5 (GID 5). Этот идентификатор будет использоваться позднее для группы `tty`. Вместо имени группы, используется идентификатор, потому что в хост-системе, может использоваться иной идентификатор для группы `tty`.

mode=0620

Это гарантирует, что все созданные devpts узлы будут иметь режим 0620 (чтение пользователем, запись пользователем, запись группой). Одновременно с вышеуказанной командой такой подход гарантирует что devpts создаст узлы устройств в соответствии с требованиями grantpt (), имея в виду вспомогательную библиотеку Glibc **pt_chown** (по умолчанию не установлена) которая не требуется.

В некоторых хост-системах, каталог `/dev/shm` является символической ссылкой на `/run/shm` . /run tmpfs был установлен выше, поэтому нужно создать только каталог.

```
if [ -h $LFS/dev/shm ]; then
  mkdir -pv $LFS/$(readlink $LFS/dev/shm)
fi
```

6.3. Управление пакетами

Часто задаваемый вопрос по книге LFS - управление пакетами. Менеджер пакетов позволяет отслеживать установку файлов, делая процесс удаления и обновления пакетов существенно проще. Кроме файлов и библиотек, пакетный менеджер будет управлять установкой файлов конфигурации. Не удивляйтесь, в этом разделе не будет информации и рекомендаций по поводу пакетного менеджера. В этом разделе, будет представлена информация о наиболее популярных механизмах работы пакетного менеджера. Идеальным пакетным менеджером для вас может стать такая программа, которая способна комбинировать несколько техник. В этом разделе также кратко упоминается о тех проблемах, которые могут возникнуть при обновлении пакетов:

- Работа с системами управления пакетов отвлекает внимание от целей этой книги - обучение тому, как построена Linux система.

- Существует множество решения, для управления пакетами. Каждый из них имеет свои достоинства и недостатки. Угодить всем - трудно.

Есть несколько советов, которые содержатся в проекте *Советы*. Ознакомьтесь с ними, возможно вы найдете решение, которое соответствует вашим потребностям.

6.3.1. Проблемы с обновлением

пакетный менеджер действительно упрощает обновление пакетов до новых версий, когда они публикуются. Как правило, инструкции в книгах LFS и BLFS могут быть использованы для обновления до новых версий. Вот некоторые моменты, которые необходимо учесть при обновлении пакетов до новых версий, особенно если обновление происходит на запущенной системе.

- Если необходимо обновить пакет Glibc до новой версии (например с 2.19 до 2.20) безопаснее всего выполнить сборку всей системы LFS заново. Также, можно выполнить пересборку всех пакетов в порядке их зависимостей. Но такой подход не рекомендуется.
- Если пакет содержащий разделяемую (shared) библиотеку обновлён, и наименование библиотеки изменилось, тогда все пакеты, которые динамически скомпонованы к библиотеке нужно заново компилировать, с указанием на новую версию библиотеки. (Обратите внимание, что нет никакой корреляции между версией пакета и имени библиотеки.). Например, рассмотрим пакет foo-1.2.3 который установил разделяемую библиотеку с наименованием `libfoo.so.1`. Допустим, вы обновили пакет до новой версии - foo-1.2.4, который установил новую версию разделяемой библиотеки `libfoo.so.2`. В данном случае, все пакеты, динамически слинкованные на библиотеку `libfoo.so.1` должны быть заново скомпилированы с указанием на новую версию библиотеки `libfoo.so.2`. Обратите внимание, что не следует удалять предыдущие версии библиотек до тех пор, пока все пакеты, которые на нёё ссылаются не перекомпилированы.

6.3.2. Методы управления пакетами

Ниже приведены некоторые общие методы управления пакетами. До принятия решения о менеджере пакетов, проведите некоторое исследование различных методов, особенно обратите внимание на их недостатки.

6.3.2.1. Все у меня в голове!

Да, это метод управления пакетами. Некоторым пользователям не нужен пакетный менеджер, потому что они прекрасно знают все установленные пакеты, и какие файлы им принадлежат. Некоторым пользователям также не требуется пакетный менеджер, потому что они пересобирают всю систему когда пакет поменяется.

6.3.2.2. Установка в отдельные каталоги

Это упрощенная техника, для которой не требуются дополнительные программы или пакеты, для управления установкой. Каждый пакет устанавливается в отдельный каталог. Например пакет foo-1.1 будет установлен в каталог `/usr/pkg/foo-1.1` и символическая ссылка будет создана с `/usr/pkg/foo` на каталог `/usr/pkg/foo-1.1`. При установке новой версии пакета foo-1.2, он будет установлен в каталог `/usr/pkg/foo-1.2` а предыдущая символическая ссылка будет заменена символической ссылкой на каталог с новой версией пакета.

Переменные окружения, такие как PATH,LD_LIBRARY_PATH ,MANPATH,INFOPATH и CPPFLAGS необходимо расширить, включив каталог /usr/pkg/foo . Для большого количества пакетов, такая техника становится неуправляемой.

6.3.2.3. Управление пакетами с использованием символических ссылок

Это вариация предыдущей техники.Каждый пакет устанавливается аналогично, но вместо создания символической ссылки, каждому файлу создаётся символическая ссылка в иерархию каталогов /usr . Это исключает необходимость модификации значений переменных окружения. Такие ссылки могут быть созданы пользователям вручную, для автоматизации создания пакетов, однако, многие менеджеры пакетов были созданы с использованием именной такого метода. Наиболее популярные из них - Stow, Epkg, Graft, and Depot.

Установка должна быть подделана, чтобы пакет считал что его установка производится в каталог /usr , однако, на самом деле, он будет установлен в иерархие каталогов /usr/pkg . Установка пакетов таким способом может быть нетривиальной задачей. Например, будет произведена установка пакета libfoo-1.1. Следующие инструкции не позволят установить пакет должным образом:

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

Установленный таким образом пакет будет работать, но те пакеты, которые имеют от него зависимости, могут не иметь ссылки на libfoo как и следовало ожидать.Если компилируется пакет, который ссылается на libfoo, можно заметить, что он связан с /usr/pkg/libfoo/1.1/lib/libfoo.so.1 вместо /usr/lib/libfoo.so.1 как и следовало ожидать. Правильный подход заключается в использовании переменной окружения DESTDIR чтобы подделать установку пакета. Такой метод работает следующим образом:

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

Большинство пакетов поддерживают такой способ, но некоторые нет. Для несовместимых пакетов, вам понадобится вручную выполнить установку пакета, или еще проще устанавливать такие пакеты в каталог /opt .

6.3.2.4. На основе временной метки

Файлу присваивается метка времени, перед установкой пакета. После установки, выполняется команда **find** с соответствующими параметрами, результат выполнения которой будет представлять из себя журнал со всеми файлами установленных после указанной метки времени. Пакетный менеджер, использующий такой подход имеет журнал установки.

Этот метод имеет преимущество - простота, но имеет и несколько недостатков. В процессе установки, файлы, которые были установленны с другими метками времени, которые отличаются от текущего времени, не будут отслеживаться пакетным менеджером. Также, возможно устанавливать один пакет за раз. Журналы ненадежны, если два пакета установлены их двух разных терминалов.

6.3.2.5. Отслеживание сценариев установки

Этот метод заключается в записи команд, выполняемых сценарием установки. Есть два подхода, как использовать данный метод:

Переменная окружения LD_PRELOAD может быть указана на предварительно загруженную библиотеку перед установкой. В процессе установки эта библиотека отслеживает пакеты, которые будут установлены присоединяя себя к различным исполняемым файлам, таким как **cp**,**install**, **mv** для отслеживания системных вызовов, которые вносят изменения в файловую систему. Для работоспособности этого метода, все исполняемые файлы должны быть динамически связаны без использования битов suid или sgid. Предзагрузка библиотеки может вызвать нежелательные побочные эффекты в процессе установки. Поэтому, рекомендуется выполнить тесты, для того, чтобы гарантировать, что пакетный менеджер не испортил что-либо и отследил все необходимые файлы.

Второй подход заключается в использовании программы **strace**, которая регистрирует все системные вызовы, во время выполнения сценариев установки.

6.3.2.6. Создание архивов для пакетов

При этой схеме, установка будет подменена в отдельное дерево каталогов, как описано в разделе Управление пакетами с использованием символических ссылок. После установки, архив с пакетом создается используя установленные файлы. Этот архив теперь будет использоваться для установки пакета либо на локальном компьютере, либо может использоваться на других машинах.

Этот подход используется большинством пакетных менеджеров в коммерческих дистрибутивах. Примеры пакетных менеджеров которые следуют этому подходу - RPM (который, кстати, требуется в базовой спецификации *Linux Standard Base Specification (LSB)*),pkg-utils, apt дистрибутива Debian и система портов Gentoo. Подсказка, описывающая, как принять этот стиль управления пакетами для систем LFS, находится по адресу *https://linuxfromscratch.ru/hints/downloads/files/fakeroot.txt*.

Создание файлов пакетов, содержащих информацию о зависимостях, является сложным и выходит за рамки LFS.

Дистрибутив Slackware использует основанную на **tar** систему для архивации пакетов. Эта система намеренно не обрабатывает зависимости пакетов как это делают более сложные менеджеры пакетов. Подробнее об управлении пакетов в Slackware см.*http://www.slackbook.org/html/package-management.html*.

6.3.2.7. Пользовательское управление пакетами.

Эта схема является уникальной для LFS была разработана Матиасом Бенкманом, и описание доступно по ссылке *Hints Project*. Суть схемы в том, что каждый пакет, будет установлен как отдельный пользователь в стандартном местоположении. Файлы, принадлежащие пакету легко идентифицируются, путём определения пользовательского идентификатора. Особенности и недостатки этого подхода слишком сложны для описания в этом разделе. Подробнее см. *https://linuxfromscratch.ru/hints/downloads/files/more_control_and_pkg_man.txt*.

6.3.3. Развертывание LFS и распространение

Одно из преимуществ системы LFS является то, что нет файлов, у которых есть строгая привязка к местоположению на диске. Можно запаковать корневой раздел (около 250MB в несжатом виде для базовой сборки LFS), например программой **tar**, скопировать по сети или записать на компакт-диск. А затем выполнить распаковку в требуемое место, и выполнить конфигурацию некоторых файлов, для правильного функционирования системы. Файлы, которые потерубется изменить включают в себя: `/etc/hosts` , `/etc/fstab` , `/etc/passwd` , `/etc/group` , `/etc/shadow` , `/etc/ld.so.conf` , `/etc/sysconfig/rc.site` , `/etc/sysconfig/network` , and `/etc/sysconfig/ifconfig.eth0` .

Может потребоваться дополнительная модификация ядра, в зависимости от разницы в аппаратной составляющей.

Note

Были сообщения о проблемах при копировании между аналогичными, но не идентичными архитектурами. Например, набор инструкций системы Intel не идентичен AMD, и версии некоторых процессоров могут иметь инструкции, которые недоступны в более ранних версиях.

Наконец, в главе Section 8.4, “Использование GRUB для настройки процесса загрузки” новая система станет загрузочной.

6.4. Вход в окружение Chroot

Теперь, необходимо войти в окружение Chroot, для начала сборки и установки окончательной системы LFS. Как пользователь `root` выполните следующую команду, для входа, с использованием созданного ранее набора временных инструментов, находящихся во временном каталоге tools:

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

Аргумент *`-i`* передаёт команде **env** значение, которое указывает полностью очистить все переменные окружения в окружении chroot. После чего, только переменные окружения HOME, TERM, PS1 и PATH будут указаны заново. Параметр *TERM=$TERM* устанавливает значение переменной окружения за пределами chroot окружения. Эта переменная необходима таким программам как **vim** и **less** для корректной работы. Если необходимы другие переменные окружения, такие как CFLAGS или CXXFLAGS, это как раз то место, где их можно указать.

Теперь нет необходимости использовать переменную окружения LFS, потому что вся работа будет ограничена файловой системой LFS. Так происходит, потому что оболочке Bash ранее сообщался в значении переменной $LFS путь до корня файловой системы, но на данный момент, она является корневым каталогом (/).

Необходимо обратить внимание что в переменной окружения PATH путь `/tools/bin` записан в последнюю очередь. Это означает, что временный инструмент не будет более использоваться после установки его окончательной версии. Это происходит, когда оболочка не “запоминает” пути к исполняемым бинарным файлам— по этой причине, хэширование выключено, указанием параметра *+h* к программе **bash**.

Когда оболочка запустится, **bash** выдаст приглашение командной строки вида `I have no name!`. Это нормально, потому что файл `/etc/passwd` ещё не создан.

Note

Очень важно чтобы все команды в этой и последующих главах запускались из среды chroot. Если был сделан выход из среды chroot по каким-либо причинам (перезагрузка, например), необходимо убедиться в том, что виртуальные файловые системы были заново примонтированы, как рассказано в главе Section 6.2.2, “Монтирование и заполнение каталога /dev” и Section 6.2.3, “Монтирование виртуальных файловых систем ядра”. Далее войдите в окружение chroot снова, чтобы продолжить установку.

6.5. Создание каталогов

Самое время создать структуру каталогов для новой системы LFS. Необходимо выполнить следующие команды, чтобы создать стандартное дерево каталогов:

```
mkdir -pv /{bin,boot,etc/{opt,sysconfig},home,lib/firmware,mnt,opt}
mkdir -pv /{media/{floppy,cdrom},sbin,srv,var}
install -dv -m 0750 /root
install -dv -m 1777 /tmp /var/tmp
mkdir -pv /usr/{,local/}{bin,include,lib,sbin,src}
mkdir -pv /usr/{,local/}share/{color,dict,doc,info,locale,man}
mkdir -v  /usr/{,local/}share/{misc,terminfo,zoneinfo}
mkdir -v  /usr/libexec
mkdir -pv /usr/{,local/}share/man/man{1..8}
mkdir -v  /usr/lib/pkgconfig

case $(uname -m) in
 x86_64) mkdir -v /lib64 ;;
esac

mkdir -v /var/{log,mail,spool}
ln -sv /run /var/run
ln -sv /run/lock /var/lock
mkdir -pv /var/{opt,cache,lib/{color,misc,locate},local}
```

По умолчанию, созданные каталоги имеют права 755, но это нежелательно для всех каталогов. В командах приведенных выше, сделаны два изменения. Первое для каталога пользователя `root`, и второе для каталогов временных файлов.

Первое изменение гарантирует, что не каждый может войти в каталог `/root` так же, как обычный пользователь будет работать со своим домашним каталогом. Следующее изменение гарантирует что любой пользователь может записывать в дерево каталогов `/tmp` , но не может удалять файлы других пользователей. Последнее запрещено так называемым “sticky bit,” - старший бит (1) в бит-маске 1777.

6.5.1. Примечания по поводу соответствия FHS

Дерево каталогов основано на стандарте иерархий файловой системы (Filesystem Hierarchy Standard,FHS). (спецификация доступна по ссылке *https://refspecs.linuxfoundation.org/fhs.shtml*). В FHS также указано необязательное наличие некоторых каталогов, таких как `/usr/local/games` and `/usr/share/games` . В этой книге будут созданы только необходимые каталоги. Однако вы свободны в ваших действиях, и можете создать эти каталоги в том числе.

6.6. Создание основных файлов и символических ссылок

Некоторые программы используют жестко зашитые пути к другим программам, которые еще не установлены. Чтобы скорректировать этот недостаток, требуется создать ряд символических ссылок, которые будут заменены реальными файлами в процессе установки программ в этой главе.

```
ln -sv /tools/bin/{bash,cat,chmod,dd,echo,ln,mkdir,pwd,rm,stty,touch} /bin
ln -sv /tools/bin/{env,install,perl,printf}         /usr/bin
ln -sv /tools/lib/libgcc_s.so{,.1}                  /usr/lib
ln -sv /tools/lib/libstdc++.{a,so{,.6}}             /usr/lib

ln -sv bash /bin/sh
```

Описание каждоый ссылки

`/bin/bash`
Большинство **bash** сценариев указывают на `/bin/bash` .

`/bin/cat`
Этот путь жестко закодирован в сценарии конфигурирования Glibc.

`/bin/dd`
Путь к `dd` будет жестко закодирован в утилите `/usr/bin/libtool`

`/bin/echo`
Используется для тестирования Glibc. Наборам тестов требуется `/bin/echo` .

`/usr/bin/env`
Этот путь жестко зашит в процедуры сборки некоторых пакетов

`/usr/bin/install`
Путь к `install` будет жестко закодирован в файле `/usr/lib/bash/Makefile.inc`

`/bin/ln`
Путь к `ln` будет жестко закодирован в файле `/usr/lib/perl5/5.30.0/<target-triplet>/Config_heavy.pl`

/bin/pwd
Некоторые **configure** сценарии, в частности Glibc's, имеют жестко закодированный путь

/bin/rm
Путь к rm будет жестко закодирован в файле /usr/lib/perl5/5.30.0/<target-triplet>/Config_heavy.pl file.

/bin/stty
Этот путь жестко закодирован в программе Expect, кроме того он необходим для выполнения набора тестов такими пакетами как Binutils и GCC.

/usr/bin/perl
Большинство Perl сценариев имеют жестко закодированный путь к программе **perl**

/usr/lib/libgcc_s.so{,.1}
Glibc требует эти ссылки для правильной работы библиотеки pthreads.

/usr/lib/libstdc++{,.6}
Эти ссылки необходимы для некоторых тестов в Glibc, а также для поддержки C ++ в GMP.

/bin/sh
Многие сценарии оболочки имеют жестко закодированный путь к /bin/sh .

Исторически, Linux поддерживает список примонтированных файловых систем в файле /etc/mtab . Современные ядра Linux имеют поддержку такого списка внутри себя, и предоставляют его через виртуальный каталог /proc . Для осуществления поддержки тех утилит и программ, которым нужны ссылка на /etc/mtab необходимо создать следующую символическую ссылку:

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

Для осуществления возможности авторизации пользователем root от имени “root”, должны быть соответствующие записи в в файлах /etc/passwd и /etc/group .

Необходимо создать файл /etc/passwd :

```
cat > /etc/passwd << "EOF"
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/dev/null:/bin/false
daemon:x:6:6:Daemon User:/dev/null:/bin/false
messagebus:x:18:18:D-Bus Message Daemon User:/var/run/dbus:/bin/false
nobody:x:99:99:Unprivileged User:/dev/null:/bin/false
EOF
```

Фактический пароль для root (символ “x” здесь используется только для заполнения) будет указан позднее.

Необходимо создать файл `/etc/group` :

```
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:
usb:x:14:
cdrom:x:15:
adm:x:16:
messagebus:x:18:
input:x:24:
mail:x:34:
kvm:x:61:
wheel:x:97:
nogroup:x:99:
users:x:999:
EOF
```

Созданные группы не являются частью какого-либо стандарта. Эти группы необходимы для конфигурации Udev в этой главе, и частично общей конвенцией используемой рядом существующих дистрибутивов Linux. Кроме того, некоторые наборы тестов зависят от конкретных пользователей и групп. Специфкация LSB (The Linux Standard Base, доступная по ссылке *https://wiki.linuxfoundation.org/lsb/start*) рекомендует, чтобы кроме группы root с идентификатором группы (GID), равным 0, присутствовала группа bin с GID, равным 1. Все другие имена групп и идентификаторы GID могут свободно выбираться системным администратором, поскольку хорошо написанные программы не зависят от номеров GID, а используют только имя группы.

Для того, чтобы убрать приглашение коммандной строки “I have no name!”, необходимо запустить новую командную оболочку. Когда был установлен пакет Glibc в главе Глава 5 и файл `/etc/group` был создан, имя пользователя и группы, теперь будут работать:

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

Обратите внимание на использование аргумента *+h*. Он сообщает программе **bash** не использовать собственный механизм хеширования путей. Без этого аргумента, **bash** будет запоминать пути к двоичным файлам которые выполнялись. Для того, чтобы использовать новые скомплированные пакеты, по мере их установки, аргумент *+h* будет использован на протяжении всей главы.

Программы **login**, **agetty**, и **init** (и другие) используют файлы с записями о событиях (log - файлов), например кто и когда авторизовался в системе. Однако, эти программы не смогут записывать файлы, до тех пор, пока их нет. Необходимо создать такие файлы, и назначить им правильные права:

```
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
```

Файл `/var/log/wtmp` хранит записи когда каждый пользователь выполняет авторизацию в системе. Файл `/var/log/faillog` хранит записи о неудачных попытках входа в систему. Очень полезно при проверке угроз в системе безопасности, хакерских атаках, попыток взлома методом перебора. Прочитать содержимое можно с помощью команды faillog. Файл `/var/log/btmp` хранит записи неудачных попыток входа в систему. Просто так, на всякий случай, если вы еще не догадались где следует искать следы активности взломщиков.

Note

Файл `/run/utmp` хранит записи о тех пользователях, которые на данные момент авторизованы в системе. Он создётся динамически, в процессе выполнения сценариев загрузки.

6.7. Заголовочные файлы API Linux-5.2.8

Заголовочные файлы Linux API (в linux-5.2.8.tar.xz) предоставляют API для использования его библиотекой C (Glibc).

Приблизительное время сборки: less than 0.1 SBU
Требуемое дисковое пространство: 960 MB

6.7.1. Установка заголовочных файлов Linux API

Ядро Linux должно предоставить интерфейс API (программный интерфейс приложения, интерфейс прикладного программирования) для использования их системной библиотекой Си (Пакет Glibc в LFS). Это можно сделать путём извлечения необходимых заголовочных файлов которые содержатся в архиве с исходным кодом ядра.

Убедитесь, что нет устаревших файлов и зависимостей от предыдущих компиляций. Команда, указанная ниже, выполнит более интенсивную очистку дерева исходных текстов. Иногда она является необходимой и можно выполнять эту команду после каждого наложения заплаток.

```
make mrproper
```

Теперь, необходимо извлечь видимые пользователем заголовочные файлы ядра из дерева исходного кода. Они будут помещены в промежуточный локальный каталог и скопированы в необходимое место, потому что процесс распаковки удаляет любые существующие файлы в целевом каталоге. Также имеются некоторые скрытые файлы, используемые разработчиками ядра. Такие файлы не требуются в LFS и могут быть удалены из промежуточного каталога.

```
make INSTALL_HDR_PATH=dest headers_install
find dest/include \( -name .install -o -name ..install.cmd \) -delete
cp -rv dest/include/* /usr/include
```

6.7.2. Содержимое пакета Linux API Headers

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
Установленные 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
`/usr/include/drm/*.h`	Заголовочные файлы Linux API DRM
`/usr/include/linux/*.h`	Заголовочные файлы Linux API
`/usr/include/misc/*.h`	The Linux API Miscellaneous Headers

`/usr/include/mtd/*.h`	Заголовочные файлы Linux 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

6.8. Man-pages-5.02

Пакет Man-pages содержит более чем 2,200 справочных страниц.

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

6.8.1. Установка пакета Man-pages

Установите пакет Man-pages выполнив команду:

```
make install
```

6.8.2. Содержимое пакета Man-pages

Установленные файлы: различные страницы руководств

Краткое описание

`man pages` Описание функций языка Си, важные файлы устройств и файлы конфигурации

6.9. Glibc-2.30

Пакет содержит стандартную библиотеку языка Си (GNU C Library). Эта библиотека предоставляет функции для выделения памяти, поиска каталогов, открытия и закрытия файлов, чтения и записи файлов, обработку строк, соответствия шаблонов (pattern matching), арифметические операции, и так далее.

Приблизительное время сборки: 21 SBU
Требуемое дисковое пространство: 3.3 GB

6.9.1. Установка пакета Glibc

Note

Система сборки Glibc самодостаточная и выполнит установку правильно, даже несмотря на то, что служебные файлы и компоновщик по прежнему указывают на каталог `/tools` . Перенастроить эти файлы нельзя до установки окончательного варианта пакета Glibc, потому что тесты программы autoconf завершатся с ошибкой.

Некоторые из программ пакета Glibc используют несовместимые со стандартом FHS каталоги и файлы, например каталог `/var/db` , предназначенный для хранения данных во время выполнения. Примените следующий патч (заплатку) чтобы исправить несовместимость со стандартом LSB:

```
patch -Np1 -i ../glibc-2.30-fhs-1.patch
```

Исправьте проблему, которая провяввляется в ядре linux-5.2:

```
sed -i '/asm.socket.h/a# include <linux/sockios.h>' \
   sysdeps/unix/sysv/linux/bits/socket.h
```

Создайте символическую ссылку для соблюдения стандартов LSB. Кроме того, для x86_64 создайте символическую ссылку для совместимости и правильной работы динамического загрузчика:

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

В документации к пакету Glibc рекомендуется выполнять компиляцию в отдельном каталоге:

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

Подготовьте Glibc к компиляции:

```
CC="gcc -ffile-prefix-map=/tools=/usr" \
../configure --prefix=/usr                          \
             --disable-werror                       \
             --enable-kernel=3.2                    \
             --enable-stack-protector=strong        \
     --with-headers=/usr/include            \
             libc_cv_slibdir=/lib
```

Значение опций, аргументов и параметров конфигурации:

`CC="gcc -ffile-prefix-map=/tools=/usr"`
Make GCC record any references to files in /tools in result of the compilation as if the files resided in /usr. This avoids introduction of invalid paths in debugging symbols.

`--disable-werror`
Аргумент отключает опцию -Werror передаваемую в GCC. Это необходимо сделать для корректного выполнения набора тестов.

`--enable-stack-protector=strong`
Аргумент усиливает безопасность системы, добавляя дополнительный код для проверки на переполнение буфера, например при переполнении буфера в стеке (Stack smashing).

`--with-headers=/usr/include`
Аргумент сообщает системе сборке местоположение заголовочных файлов API ядра. По умолчанию поиск заголовков будет в каталоге `/tools/include` .

`libc_cv_slibdir=/lib`
Аргумент указывает корректную библиотеку для всех архитектур. Нет необходимости использовать lib64.

Скомпилируйте пакет:

```
make
```

Important

В этом разделе выполнение наборов тестов пакета Glibc особенно важно. Не пропускайте их выполнение ни при каких обстоятельствах.

Как правило, некоторые тесты завершатся с ошибкой. Тесты, завершенные с ошибкой и перечисленные ниже, можно игнорировать.

```
case $(uname -m) in
  i?86)   ln -sfnv $PWD/elf/ld-linux.so.2        /lib ;;
  x86_64) ln -sfnv $PWD/elf/ld-linux-x86-64.so.2 /lib ;;
esac
```

Note

Символическая ссылка созданная выше необходима для запуска тестов на этом этап сборки в среде chroot. Позднее, эта ссылка будет перезаписана на следующем этапе.

```
make check
```

Можно наблюдать некоторые ошибки при тестировании. Выполнение набора тестов Glibc сильно зависит от хост-системы. Ниже приведен список наиболее распространённых проблем, наблюдаемых в некоторых версиях LFS:

- *misc/tst-ttyname* завершается с ошибкой в среде chroot LFS.
- *inet/tst-idna_name_classify* завершается с ошибкой в среде chroot LFS.
- *posix/tst-getaddrinfo4* and *posix/tst-getaddrinfo5* на некоторых архитектурах может завершиться с ошибкой.
- *nss/tst-nss-files-hosts-multi* может завершиться с ошибкой по неизвестным причинам.
- *rt/tst-cputimer{1,2,3}* выполнение тестов зависит от версии ядра хост системы. Версии 4.14.91–4.14.96,4.19.13–4.19.18, and 4.20.0–4.20.5 - как известн, вызовут ошибки в этих тестах.
- Математические тесты иногда не проходят, когда они запускаются на относительно старых процессорах Intel или AMD.

Хотя это и безобидное сообщение, но на этапе установки Glibc будет выдавать предупреждение об отсутствии файла `/etc/ld.so.conf` . Предотвратите это предупреждение, выполнив команду:

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

Исправьте сгенерированный Makefile чтобы пропустить ненужные проверки на корректность, которые могут завершиться неудачно в среде LFS:

```
sed '/test-installation/s@$(PERL)@echo not running@' -i ../Makefile
```

Установите пакет:

```
make install
```

Установите конфигурационный файл и выполните команду time directory для программы **nscd**:

```
cp -v ../nscd/nscd.conf /etc/nscd.conf
mkdir -pv /var/cache/nscd
```

Далее, установите локали для обеспечения реакции системы на изменение языков. Ни одна из локалей не требуется, но если некоторые из них не будут установлены, при выполнении тестов в следующих пакетах важные проверки могут быть пропущены.

Отдельные локали можно установить с помощью программы localedef. Например, первая команда localedef ниже объединяет определение /usr/share/i18n/locales/cs_CZ без набора символов с помощью определения /usr/share/i18n/charmaps/UTF-8.gz charmap и добавляет результат в /usr/lib/locale/locale-archive. В следующих инструкциях будет установлен минимальный набор локалей, необходимых для оптимального покрытия тестов:

```
mkdir -pv /usr/lib/locale
localedef -i POSIX -f UTF-8 C.UTF-8 2> /dev/null || true
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 UTF-8 en_GB.UTF-8
localedef -i en_HK -f ISO-8859-1 en_HK
localedef -i en_PH -f ISO-8859-1 en_PH
localedef -i en_US -f ISO-8859-1 en_US
localedef -i en_US -f UTF-8 en_US.UTF-8
localedef -i es_MX -f ISO-8859-1 es_MX
localedef -i fa_IR -f UTF-8 fa_IR
localedef -i fr_FR -f ISO-8859-1 fr_FR
localedef -i fr_FR@euro -f ISO-8859-15 fr_FR@euro
localedef -i fr_FR -f UTF-8 fr_FR.UTF-8
localedef -i it_IT -f ISO-8859-1 it_IT
localedef -i it_IT -f UTF-8 it_IT.UTF-8
localedef -i ja_JP -f EUC-JP ja_JP
localedef -i ja_JP -f SHIFT_JIS ja_JP.SIJS 2> /dev/null || true
localedef -i ja_JP -f UTF-8 ja_JP.UTF-8
localedef -i ru_RU -f KOI8-R ru_RU.KOI8-R
localedef -i ru_RU -f UTF-8 ru_RU.UTF-8
localedef -i tr_TR -f UTF-8 tr_TR.UTF-8
localedef -i zh_CN -f GB18030 zh_CN.GB18030
localedef -i zh_HK -f BIG5-HKSCS zh_HK.BIG5-HKSCS
```

Дополнительно, можно установить локаль Вашей страны, языка и набора символов

Вместо всего вышеизложенного, можно выполнить установку всех локалей за раз, представленных в файле `glibc-2.30/localedata/SUPPORTED` (он содержит все локали, перечисленные выше, и многие другие). Выполните следующую команду:

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

Затем, используйте команду **localedef** чтобы создать и установить те локали, которые не представленны в файле `glibc-2.30/localedata/SUPPORTED` если они необходимы.

Note

Glibc now uses libidn2 when resolving internationalized domain names. This is a run time dependency. If this capability is needed, the instrucions for installing libidn2 are in the *BLFS libidn2 page*.

6.9.2. Настройка Glibc

6.9.2.1. Добавление nsswitch.conf

Файл `/etc/nsswitch.conf` необходимо создать потому что настройки Glibc по умолчанию не будут правильно работать в сетевой (networked) среде.

Создайте новый файл `/etc/nsswitch.conf` выполнив следующую команду:

```
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
```

6.9.2.2. Добавление данных о часовых поясах

Установите и настройте данные о часовых поясах выполнив следующую команду:

```
tar -xf ../../tzdata2019b.tar.gz

ZONEINFO=/usr/share/zoneinfo
mkdir -pv $ZONEINFO/{posix,right}

for tz in etcetera southamerica northamerica europe africa antarctica  \
          asia australasia backward pacificnew systemv; 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 zic-команд:

zic -L /dev/null ...
Создаёт POSIX временные зоны, без каких-либо секунд. Это условие для того, чтобы положить их в zoneinfo и zoneinfo/posix . Необходимо положить POSIX временные зоны в файл zoneinfo, иначе некоторые тесты будут завершаться с ошибками. На встраиваемых системах, где мало дискового пространства и нет необходимости обновлять данные о часовых поясах, можно сохранить около 1.9 МБ не используя каталог posix но тогда, некоторые приложения могут работать с ошибками а тесты могут не проходить.

zic -L leapseconds ...
Создаёт правильные временные зоны, включая секунды. На встраиваемых системах, где мало дискового пространства и нет необходимости обновлять данные о часовых поясах и заботиться о правильном времени, можно сохранить около 1.9 МБ не используя каталог опустив каталог right.

zic ... -p ...
Создаёт файл posixrules file. Мы используем New York, потому что POSIX требует соблюдения правил летнего времени в соответствии с правилами США.

Один из способов определить местный часовой пояс - запустить следующие сценарий:

```
tzselect
```

После ответов не вопросы о местоположении, сценарий выдаст наименование временной зоны (например *America/Edmonton*). Есть и другие часовые пояса, которые указанны в файле: /usr/share/zoneinfo , такие как *Canada/Eastern* или *EST5EDT* котрые не распознаются запущеным сценарием, но могут быть использованы.

Создайте файл /etc/localtime выполнив команду:

```
ln -sfv /usr/share/zoneinfo/<xxx> /etc/localtime
```

Замените *<xxx>* наименованием выбранной временной зоны (например, Canada/Eastern).

6.9.2.3. Конфигурирование динамического загрузчика

По умолчанию, динамический загрузкчик (/lib/ld-linux.so.2) выполняет поиск динамических библиотек в каталогах /lib и /usr/lib которые необходимы для запущенных программ. Однако, если есть каталоги, в которых содержаться динамические библиотеки, и эти каталоги отличаются от вышеуказанных, их необходимо добавить в файл /etc/ld.so.conf в том порядке, в котором необходимо, чтобы динамический загрузчик выполнял поиск. Есть ещё два известных каталога, где могу содержаться динамические библиотеки: /usr/local/lib и /opt/lib , поэтому, можно добавить эти каталоги в пути поиска библиотек динамического загрузчика.

Создайте новый файл /etc/ld.so.conf выполнив следующую команду:

```
cat > /etc/ld.so.conf << "EOF"
# Begin /etc/ld.so.conf
/usr/local/lib
/opt/lib

EOF
```

При желании динамический загрузчик может также выполнять поиск в каталоге включая содержимое найденных там файлов. Обычно файлы в этом каталоге включает одну строку, указывающую нужный путь библиотеки. 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
```

6.9.3. Содержимое пакета Glibc

Установленные программы: catchsegv, gencat, getconf, getent, iconv, iconvconfig, ldconfig, ldd, lddlibc4, locale, localedef, makedb, mtrace, nscd, pcprofiledump, pldd, sln, sotruss, sprof, tzselect, xtrace, zdump, and zic

Установленные библиотеки: ld-2.30.so, libBrokenLocale.{a,so}, libSegFault.so, libanl.{a,so}, libc.{a,so}, libc_nonshared.a, libcrypt.{a,so}, libdl.{a,so}, libg.a, libm.{a,so}, libmcheck.a, libmemusage.so, libmvec.{a,so}, libnsl.{a,so}, libnss_compat.so, libnss_dns.so, libnss_files.so, libnss_hesiod.so, libpcprofile.so, libpthread.{a,so}, libpthread_nonshared.a, libresolv.{a,so}, librt.{a,so}, libthread_db.so, and libutil.{a,so}

Установленные 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, /var/cache/nscd, and /var/lib/nss_db

Краткое описание

catchsegv Может использоваться для создания трассировки стека, когда программа завершается с ошибкой сегментации

gencat Создает каталоги сообщений

getconf Отображает значения конфигурации системы для специфичных переменных файловой системы

getent Получает записи из административной базы данных

iconv Выполняет преобразование набора символов

iconvconfig Создает ускоренную загрузку **iconv** модулей файлов конфигурации

ldconfig Настраивает привязки динамического компоновщика

ldd помогает определить список разделяемых библиотек (shared libraries), от которых зависит программа.

lddlibc4 Помогает **ldd** с объектными файлами

locale Отображает всевозможную информацию о текущей локали

localedef	Компилирует спецификации локали
makedb	Создает простую базу данных из текстового ввода
mtrace	Читает и интерпретирует файл трассировки памяти и отображает сводку в удобочитаемом формате
nscd	Служба (демон), которая предоставляет кэш для наиболее общих запросов службы имен.
pcprofiledump	Dump information generated by PC profiling
pldd	Список динамических общих объектов, используемых запущенными процессами
sln	Статически слиокнванные **ln** программы
sotruss	Выполняет трассировку вызовов процедуры разделяемой библиотеки для указанной команды
sprof	Считывает и отображает данные профилирования общих объектов
tzselect	Выясняет у пользователя его текущее местоположение и выводит описание часового пояса на устройство стандартного вывода.
xtrace	Трассировка выполняемой программы, и выводит в реальном времени на устройство стандартного вывода выполняемые функции
zdump	распечатывает текущее время для каждого часового пояса, указанного в командной строке
zic	компилятор часовых поясов
`ld-2.30.so`	Программа выполняет поиск и загружают динамические библиотеки, необходимые программам, а также подготавливают программы к запуску и запускают их.
`libBrokenLocale`	Используется внутри Glibc как грубый хак, чтобы обработать запущенную сломанную программу (например некоторые приложения Motif). Изучите комментарии в файле `glibc-2.30/locale/broken_cur_max.c` для получения более подробной информации
`libSegFault`	Обработчик сигнала ошибки сегментации, используемый **catchsegv**
`libanl`	Асинхронная библиотека поиска имен
`libc`	Стандартная библиотека языка Си
`libcrypt`	Криптографическая библиотека
`libdl`	Интерфейс библиотеки динамической линковки
`libg`	Заглушка-библиотека, не содержащая функций. Раньше была библиотеки выполнения для **g++**
`libm`	Математическая библиотека
`libmcheck`	Включает проверку распределения памяти при линковке
`libmemusage`	Используется программой **memusage** чтобы помочь собрать информацию об использовании памяти в программе
`libnsl`	Библиотека сетевых сервисов

`libnss`	Библиотеки коммутаторов имен, содержащие функции для разрешение имен хостов, имен пользователей, имен групп, псевдонимов, служб, протоколов и т.д.
`libpcprofile`	Can be preloaded to PC profile an executable
`libpthread`	POSIX библиотека потоков
`libresolv`	Содержит функции для создания, отправки и интерпретации пакетов на серверы доменных имен в Интернете
`librt`	Содержит функции, обеспечивающие большую часть указанных интерфейсов в POSIX.1b расширении
`libthread_db`	Содержит функции, полезные для построения отладчиков для многопоточных программ
`libutil`	Содержит кл для “стандартных” функций, используемых в большинстве различных утилит Unix

6.10. Перенастройка временного набора инструментов

Теперь, когда все библиотеки Си были установлены, необходимо перенастроить используемый сейчас временный набор инструментов, чтобы он выполнял линковку новых скомпилированых программ с установленными библиотеками.

Сначала,выполните резервное копирование компоновщика `/tools` , и замените его настроенным компоновщиком в главе 5. Также будут созданы ссылки на их копии в каталоге `/tools/$(uname -m)-pc-linux-gnu/bin` :

```
mv -v /tools/bin/{ld,ld-old}
mv -v /tools/$(uname -m)-pc-linux-gnu/bin/{ld,ld-old}
mv -v /tools/bin/{ld-new,ld}
ln -sv /tools/bin/ld /tools/$(uname -m)-pc-linux-gnu/bin/ld
```

Далее, необходимо изменить специальные файлы GCC так, чтобы они указывали на новый динамический компоновщик. После удаления всех экземпляров в каталоге “/tools” останутся только правильные пути к динамическому компоновщику. Также необходимо настроить некоторые файлы GCC для правильного поиска заголовочных файлов и файлов запуска Glibc. С помощью команды **sed** можно выполнить следующую команду:

```
gcc -dumpspecs | sed -e 's@/tools@@g'                   \
    -e '/\*startfile_prefix_spec:/{n;s@.*@/usr/lib/ @}' \
    -e '/\*cpp:/{n;s@$@ -isystem /usr/include@}' >      \
    `dirname $(gcc --print-libgcc-file-name)`/specs
```

Хорошей идеей будет выполнить проверку сделанных изменений на предмет корректности работы.

На этом этапе следует проверить и убедиться в том, что базовые функции (линковка и компиляция) перенастроенного временного набора инструментов работают так, как необходимо. Выполним такие проверки:

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

В результате выполнения этой команды не должно быть ошибок, и вывод (в зависимости от платформы) должен соответствовать следующей строке:

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

Обратите внимание, что для 64-битной системы каталог `/lib` это место, где располагается динамический компоновщик, но доступ к нему обеспечен через символическую ссылку /lib64.

Note

Для 32-битных систем интерпретатор должен быть /lib/ld-linux.so.2.

Теперь убедимся что настройка стартовых файлов запуска выполнена правильно:

```
grep -o '/usr/lib.*/crt[1in].*succeeded' dummy.log
```

Вывод должен соответствовать следующей строке:

```
/usr/lib/../lib/crt1.o succeeded
/usr/lib/../lib/crti.o succeeded
/usr/lib/../lib/crtn.o succeeded
```

Проверьте, что компилятор выполняет поиск заголовочных файлов в нужных местах:

```
grep -B1 '^ /usr/include' dummy.log
```

Вывод должен соответствовать следующей строке:

```
#include <...> search starts here:
 /usr/include
```

Далее проверим, что компоновщик использует правильные пути поиска:

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

Ссылки на пути с наличием у компонента суффикса '-linux-gnu' должны быть проигнорированы, и вывод должен соответствовать следующей строке:

```
SEARCH_DIR("/usr/lib")
SEARCH_DIR("/lib")
```

Проверим что используется нужная библиотека libc:

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

Вывод должен соответствовать следующей строке:

```
attempt to open /lib/libc.so.6 succeeded
```

Наконец, проверим что GCC использует правильный динамический компоновщик:

```
grep found dummy.log
```

Вывод (в зависимости от платформы в наименовании динамического компоновщика) должен соответствовать следующей строке:

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

Если вывод на выполнение команд не выводиться или команды не выполняются вовсе, то это признак того, что в предыдущих инструкциях была допущена серьезная ошибка. Исследуйте предыдущие шаги и проверьте правильность их выполнения. Постарайтесь найти ошибку в вводимых командах. Наиболее вероятно, что причина в проблемах - не правильная настройка специальных файлов при перенастройке временного набора инструментов. Все проблемы должны быть найдены и исправлены здесь, до выполнения следующих этапов.

Если все работает правильно, выполним очистку тестовых файлов:

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

6.11. Zlib-1.2.11

Пакет Zlib содержит библиотеку для сжатия и распаковки, которую используют некоторые программы.

Приблизительное время сборки: less than 0.1 SBU
Требуемое дисковое пространство: 5.1 MB

6.11.1. Установка пакета Zlib

Подготовьте пакет Zlib к компиляции:

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

Скомпилируйте пакет:

```
make
```

Для выполнения тестов, выполните команду:

```
make check
```

Установите пакет:

```
make install
```

Разделяемые библиотеки нужно переместить в `/lib` , и в результате `.so` файл в `/usr/lib` нужно будет создать заново:

```
mv -v /usr/lib/libz.so.* /lib
ln -sfv ../../lib/$(readlink /usr/lib/libz.so) /usr/lib/libz.so
```

6.11.2. Содержимое пакета Zlib

Установленные библиотеки: libz.{a,so}

Краткое описание

`libz` Содержит функции для сжатия и распаковки, которые используют некоторые программы

6.12. File-5.37

Пакет содержит утилиты для определения типов файлов. Некоторым пакетам требуется, чтобы этот пакет был установлен.

Приблизительное время сборки: 0.1 SBU
Требуемое дисковое пространство: 19 MB

6.12.1. Установка пакета File

Подготовьте пакет File к компиляции:

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

Скомпилируйте пакет:

```
make
```

Для выполнения тестов, выполните команду:

```
make check
```

Установите пакет:

```
make install
```

6.12.2. Содержимое пакета File

Установленные программы: file
Установленная библиотека: libmagic.so

Краткое описание

file Пытается классифицировать каждый переданный файл; он делает это, выполняя несколько тестов: тесты файловой системы, тесты магических чисел и языковые тесты

`libmagic` Подпрограммы для распознавания магических чисел, используемые программой **file**

6.13. Readline-8.0

Пакет Readline - набор библиотек, который предлагает редактирование командной строки и возможности просмотра истории.

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

6.13.1. Установка пакета Readline

Для установки рабочего пакета Readline, необходимо переместить старые библиотеки в <libraryname>.old. Как правило проблем не будет, однако иногда могут возникать ошибки привязки с **ldconfig**. Это можно обойти, выполнив следующие команды sed:

```
sed -i '/MV.*old/d' Makefile.in
sed -i '/{OLDSUFF}/c:' support/shlib-install
```

Подготовьте пакет Readline к компиляции:

```
./configure --prefix=/usr    \
            --disable-static \
            --docdir=/usr/share/doc/readline-8.0
```

Скомпилируйте пакет:

```
make SHLIB_LIBS="-L/tools/lib -lncursesw"
```

Значение аргументов команды make:

SHLIB_LIBS="-L/tools/lib -lncursesw"
Аргумент сообщает выполнить принудительную линковку с библиотекой `libncursesw`.

У этого пакета нет тестов.

Установите пакет:

```
make SHLIB_LIBS="-L/tools/lib -lncurses" install
```

Теперь переместите динамические библиотеки в подходящее место и исправьте символические ссылки:

```
mv -v /usr/lib/lib{readline,history}.so.* /lib
chmod -v u+w /lib/lib{readline,history}.so.*
ln -sfv ../../lib/$(readlink /usr/lib/libreadline.so) /usr/lib/libreadline.so
ln -sfv ../../lib/$(readlink /usr/lib/libhistory.so ) /usr/lib/libhistory.so
```

При желании, можно установить документацию:

```
install -v -m644 doc/*.{ps,pdf,html,dvi} /usr/share/doc/readline-8.0
```

6.13.2. Содержимое пакета Readline

Установленные библиотеки: libhistory.so и libreadline.so

Установленные каталоги: /usr/include/readline и /usr/share/doc/readline-8.0

Краткое описание

`libhistory` Обеспечивает согласованный пользовательский интерфейс для вызова строк истории

`libreadline` Предоставляет набор команд для манипулирования текстом, введенным в интерактивной сессии программы.

6.14. M4-1.4.18

Пакет содержит общий макропроцессор текста - полезный инструмент для выполнения сборки других программ.

Приблизительное время сборки: 0.4 SBU
Требуемое дисковое пространство: 33 MB

6.14.1. Установка пакета M4

Сначала, сделаем некоторые исправления, требуемые glibc-2.28:

```
sed -i 's/IO_ftrylockfile/IO_EOF_SEEN/' lib/*.c
echo "#define _IO_IN_BACKUP 0x100" >> lib/stdio-impl.h
```

Подготовьте пакет M4 к компиляции:

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

Скомпилируйте пакет:

```
make
```

Для выполнения тестов, выполните команду:

```
make check
```

Установите пакет:

```
make install
```

6.14.2. Содержимое пакета M4

Установленная программа: m4

Краткое описание

m4 Копирует данные файлы при расширении макросов, которые они содержат. Эти макросы являются встроенными или определяемыми пользователем и могут принимать любое количество аргументов. Помимо выполнения макрорасширения, m4 имеет встроенные функции для включения именованных файлов, выполнения команд Unix, выполнения целочисленной арифметики, обработки текста, рекурсии и т. Д. Программа m4 может использоваться либо как фронт-енд для компилятора, или как самостоятельный макропроцессор.]

6.15. Bc-2.1.3

BC (basic calculator) — интерактивный интерпретатор Си-подобного языка, позволяет выполнять вычисления с произвольно заданной точностью. Часто используется как калькулятор в командной строке UNIX-подобных операционных систем.

Приблизительное время сборки: 0.1 SBU
Требуемое дисковое пространство: 2.8 MB

6.15.1. Установка пакета Bc

Подготовьте пакет Bc к компиляции:

```
PREFIX=/usr CC=gcc CFLAGS="-std=c99" ./configure.sh -G -O3
```

Значение параметров конфигурации:

CC=gcc CFLAGS="-std=c99"
Эти параметры указывают компилятор и стандарт Си для использования.

-O3
Определяют используемую оптимизацию. Символы отладки будут сохранены.

-G
Исключает тесты, которые не работают без GNU bc.

Скомпилируйте пакет:

```
make
```

Выполните тесты bc, выполнив:

```
make test
```

Установите пакет:

```
make install
```

6.15.2. Содержимое пакета Bc

Установленные программы: bc and dc

Краткое описание

bc Калькулятор командной строки

dc Калькулятор командной строки с обратной обработкой

6.16. Binutils-2.32

Пакет содержит компоновщик, ассемблер, и другие утилиты и инструменты для работы с объектными файлам. Программы в этом пакете необходимы для компиляции как большинства пакетов системы LFS, так и многих пакетов за её пределами.

Приблизительное время сборки: 7.4 SBU
Требуемое дисковое пространство: 5.1 GB

6.16.1. Установка пакета Binutils

Проверим, что псевдотерминалы (PTY) правильно работают в среде chroot. Для этого выполним простой тест:

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

Результатом выполнения команды, на экране должна появиться следующая строка:

```
spawn ls
```

Однако, если вместо вышеуказанного результата, появилось сообщение, как показано ниже, это означает что среда не правильно настроена для работы с псевдотерминалами. Эту проблему необходимо решить до запуска тестов пакета Binutils и GCC:

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

Теперь удалите один тест, который мешает запускать тесты. до завершения:

```
sed -i '/@\tincremental_copy/d' gold/testsuite/Makefile.in
```

В документации к пакету BinUtils рекомендуется выполнять компиляцию в отдельном каталоге:

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

Подготовьте пакет Binutils к компиляции:

```
../configure --prefix=/usr       \
             --enable-gold       \
             --enable-ld=default \
             --enable-plugins    \
             --enable-shared     \
             --disable-werror    \
             --enable-64-bit-bfd \
             --with-system-zlib
```

Значение параметров конфигурации:

`--enable-gold`
Выполнять сборку gold-компоновщика и устанавливает его как ld.gold (рядом с линковщиком по умолчанию).

`--enable-ld=default`
Выполнить сборку обычного bdf компоновщика и установить его как ld (компоновщик по умолчанию) и как ld.bfd.

`--enable-plugins`
Включить поддержку плагинов для компоновщика

`--enable-64-bit-bfd`
Включение 64-битной поддержки (для хостов с ограниченным размером слова). Может не потребоваться на 64-битной системе, однако эта опция не нанесет вреда.

`--with-system-zlib`
Использовать уже установленную библиотеку zlib, вместо использования встроенной версии.

Скомпилируйте пакет:

```
make tooldir=/usr
```

Значение параметров make:

`tooldir=/usr`
Как правило, значение tooldir (каталог в котором будут располагаться исполняемые файлы) установлено как `$(exec_prefix)/$(target_alias)` . На пример, на машине x86_64 значение приобретёт вид `/usr/x86_64-unknown-linux-gnu` . Потому что это пользовательская система, и её целевой каталог в `/usr` не требуется. Путь `$(exec_prefix)/$(target_alias)` будет использоваться только в том случае, если система будет задействована в качестве кросс-компилятора (например, необходимо выполнить компиляцию пакета на машине Intel которая должна сгенерировать код, работающий на машинах PowerPC).

Important

Выполнение тестов пакета Binutils считается критичным. Не пропускайте их выполнение ни при каких обстоятельствах.

Для выполнения тестов, выполните команду:

```
make -k check
```

The PC-relative offset и the debug_msg.sh могут не пройти в окружении LFS.

Один тест, debug_msg.sh, как известно, не проходит

Установите пакет:

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

6.16.2. Содержимое пакета Binutils

Установленные программы:	addr2line, ar, as, c++filt, dwp, elfedit, gprof, ld, ld.bfd, ld.gold, nm, objcopy, objdump, ranlib, readelf, size, strings, and strip
Установленные библиотеки:	libbfd.{a,so} and libopcodes.{a,so}
Каталог установки:	/usr/lib/ldscripts

Краткое описание

addr2line Переводит адреса программ на имена файлов и номера строк; учитывая адрес и имя исполняемого файла, он использует отладочную информацию в исполняемом файле, чтобы определить, какой исходный файл и строка число связано с адресом

ar создаёт и изменяет архивы, а также извлекает файлы из них

as Ассемблер, известный как GAS (Gnu Assembler). Выполняет создание объектного файла из вывода команды **gcc**

c++filt Используется компоновщиком для удаления символов C ++ и Java и сохранения перегруженных функции от конфликтов

dwp утилита для упаковки DWARF

elfedit Обновляет ELF заголовок в файлах ELF

gprof Отображает график вызовов данных профилирования

ld Компоновщик, который объединяет ряд объектных и архивных файлов в один файл, перемещая их данные и связывает ссылки

ld.gold gold — это компоновщик для ELF файлов. Он стал официальным пакетом GNU и был добавлен в binutils в марте 2008 года и был впервые выпущен в составе binutils версии 2.19. Мотивацией для написания gold было создание компоновщика, который является более быстрым, чем GNU linker, особенно для больших приложений, написанных на C++.

ld.bfd Жесткая ссылка на **ld**

nm Перечисляет символы, встречающиеся в данном объектном файле

objcopy Копирование объектных файлов (возможно с изменениями)

objdump Отображает информацию о данном объектном файле с возможностью контроля необходимой информации для отображения; Эта информация может быть полезна для программистов, которые работают над инструментами для компиляции

ranlib Создает индекс содержимого архива и сохраняет его в архиве; индекс перечисляет все символы, определенные архивом, которые являются перемещаемыми объектными файлами

readelf показ содержимого исполняемых файлов в формате ELF

size вывод общего размера и размера секций

strings Выводит для каждого заданного файла, последовательности печатаемых символов, которые имеют, по меньшей мере, указанную длину (по умолчанию 4); для объектных файлов он печатает по умолчанию только строки из разделов инициализации и загрузки, а для других типов файлов - сканирует весь файл

strip Отбрасывает символы из объектных файлов

`libbfd` Библиотека двоичных файловых дескрипторов (Binary File Descriptor)

`libopcodes` libopcodes - библиотека для работы с опкодами, используется в сборке утилит вроде objdump. Опкоды - “читаемые” версии инструкций процессора.

6.17. GMP-6.1.2

Библиотека GMP (GNU Multi-Precision Library), предназначенная для вычислений с плавающей запятой, целыми и рациональными числами с произвольной точностью. Библиотека широко используется в криптографических целях и для компьютерных вычислений. Данная библиотека необходима для сборки gcc.

Приблизительное время сборки: 1.2 SBU
Требуемое дисковое пространство: 61 MB

6.17.1. Установка пакета GMP

Note

Если вы выполняете сборку для 32-битной машины x86,но у вас процессор, который совместим и может выполнять 64-битные инструкции и указана переменная окружения CFLAGS, сценарий конфигурирования будет запускаться для 64-битной системы и в результате процесс завершиться с ошибкой. Такого поведения можно избежать, если запустить команду configure, как указано ниже:

```
ABI=32 ./configure ...
```

Note

Значения по умолчанию GMP создают библиотеки оптимизированные для процессора хост-системы. Если библиотеки, подходящие для процессоров, не желательны, чем процессор хост-системы, то общие библиотеки могут быть созданы следующим образом:

```
cp -v configfsf.guess config.guess
cp -v configfsf.sub   config.sub
```

Подготовьте пакет GMP к компиляции:

```
./configure --prefix=/usr    \
            --enable-cxx     \
            --disable-static \
            --docdir=/usr/share/doc/gmp-6.1.2
```

Значения новых параметров конфигурации:

--enable-cxx
Аргумент включает поддержку языка Си++

--docdir=/usr/share/doc/gmp-6.1.2
Значение этого аргумента указывает на правильное местоположение документации

Выполните компиляцию пакета, и сгенерируйте HTML документацию:

```
make
make html
```

Important

Выполнение тестов пакета GMP считается критичным. Не пропускайте их выполнение ни при каких обстоятельствах.

Для выполнения тестов, выполните команду:

```
make check 2>&1 | tee gmp-check-log
```

Caution

Код, скомпилированный пакетом очень сильно оптимизирован для того процессора, на котором он был скомпилирован. Иногда код, который считывает процессор, неверно идентифицирует возможности системы и будут возникать ошибки в тестах или других программах, использующих библиотеки gmp, с сообщением "Illegal instruction". В таком случае, необходимо выполнить повторную конфигурацию пакета с аргументом --build=x86_64-unknown-linux-gnu и выполнить сборку повторно.

убедитесь, что все 190 тестов в наборе прошли успешно. Проверьте результаты тестирования, выполнив следующую команду:

```
awk '/# PASS:/{total+=$3} ; END{print total}' gmp-check-log
```

Установите пакет и его документацию:

```
make install
make install-html
```

6.17.2. Содержимое пакета GMP

Установленные библиотеки: libgmp.so and libgmpxx.so
Каталог установки: /usr/share/doc/gmp-6.1.2

Краткое описание

`libgmp` Содержит функции предназначенные для вычислений с плавающей запятой, целыми и рациональными числами с произвольной точностью.

`libgmpxx` Содержит функции для языка Си++ предназначенные для вычислений с плавающей запятой, целыми и рациональными числами с произвольной точностью.

6.18. MPFR-4.0.2

Пакет MPFR включает функции по работе с вычислениями с произвольной точностью.

Приблизительное время сборки: 0.9 SBU
Требуемое дисковое пространство: 37 MB

6.18.1. Установка пакета MPFR

Подготовьте пакет MPFR к компиляции:

```
./configure --prefix=/usr        \
            --disable-static     \
            --enable-thread-safe \
            --docdir=/usr/share/doc/mpfr-4.0.2
```

Выполните компиляцию пакета, и сгенерируйте HTML документацию:

```
make
make html
```

Important

Выполнение тестов пакета MPFR считается критичным. Не пропускайте их выполнение ни при каких обстоятельствах.

Выполните команду для запуска тестов и убедитесь, что все тесты выполнились успешно:

```
make check
```

Установите пакет и его документацию:

```
make install
make install-html
```

6.18.2. Содержимое пакета MPFR

Установленные библиотеки: libmpfr.so
Каталог установки: /usr/share/doc/mpfr-4.0.2

Краткое описание

`libmpfr` Содержит функции по работе с вычислениями с произвольной точностью

6.19. MPC-1.1.0

Пакет MPC содержит функции предназначенные для вычислений с плавающей запятой, целыми и рациональными числами с произвольной точностью. Он необходим пакету GCC.

Приблизительное время сборки: 0.3 SBU
Требуемое дисковое пространство: 22 MB

6.19.1. Установка пакета MPC

Подготовьте пакет MPC к компиляции:

```
./configure --prefix=/usr    \
            --disable-static \
            --docdir=/usr/share/doc/mpc-1.1.0
```

Выполните компиляцию пакета, и сгенерируйте HTML документацию:

```
make
make html
```

Для выполнения тестов, выполните команду:

```
make check
```

Установите пакет и его документацию:

```
make install
make install-html
```

6.19.2. Содержимое пакета MPC

Установленные библиотеки: libmpc.so
Каталог установки: /usr/share/doc/mpc-1.1.0

Краткое описание

`libmpc` содержит функции предназначенные для вычислений с плавающей запятой, целыми и рациональными числами с произвольной точностью

6.20. Shadow-4.7

Пакет Shadow содержит программы для работы с паролями безопасным способом.

Приблизительное время сборки: 0.2 SBU
Требуемое дисковое пространство: 46 MB

6.20.1. Установка пакета Shadow

Note

Если вы хотите принудительно использовать надежные пароли, изучите информацию по ссылке *https://linuxfromscratch.ru/blfs/view/svn/postlfs/cracklib.html* для установки библиотеки CrackLib до того, как будет установлен пакет Shadow. Потом добавьте еще один аргумент *--with-libcrack* в сценарий конфигурирования пакета.

Запретим установку программы **groups** и её руководств, так как пакет Coreutils предоставляет более лучшую версию. Также предотвратите установку страниц руководств которые уже содержатся в пакете Section 6.8, “Man-pages-5.02”:

```
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 того, чтобы использовать метод по умолчанию *crypt*, мы будем использовать более стойкий и безопасный метод *SHA-512* для шифрования паролей, который также позволяет хранить пароли длиннее чем 8 символов. Следует также изменить устаревший путь к каталогу `/var/spool/mail`, местоположением для почтовых ящиков пользователей, которые Shadow использует по умолчанию в каталоге `/var/mail`:

```
sed -i -e 's@#ENCRYPT_METHOD DES@ENCRYPT_METHOD SHA512@' \
       -e 's@/var/spool/mail@/var/mail@' etc/login.defs
```

Note

Если вы приняли решение выполнять сборку Shadow с поддержкой Cracklib, выполните следующую команду:

```
sed -i 's@DICTPATH.*@DICTPATH\t/lib/cracklib/pw_dict@' etc/login.defs
```

Внесите незначительные изменения, чтобы создать первый номер группы, сгенерированный программой useradd равным 1000:

```
sed -i 's/1000/999/' etc/useradd
```

Подготовьте пакет Shadow к компиляции:

```
./configure --sysconfdir=/etc --with-group-name-max-length=32
```

Значения параметров конфигурации:

`--with-group-name-max-length=32`
Максимальная длина имени пользователя-32 символа. Сделаем такое же значение для групп.

Скомпилируйте пакет:

```
make
```

У этого пакета нет тестов.

Установите пакет:

```
make install
```

Переместим программу в нужное место:

```
mv -v /usr/bin/passwd /bin
```

6.20.2. Конфигурация Shadow

Пакет содержит утилиты для добавления, модификации и удаления пользователей и групп; установки и изменения паролей; а также выполняет другие задачи по администрированию. Для полного разъяснения что означает *password shadowing*, обратитесь к документации, которая расположена в каталоге `doc/HOWTO` в архиве пакета. Если вы используете функциональную поддержку Shadow, необходимо держать в уме что программы, которым необходима проверка паролей (дисплейные менеджеры, программы FTP, pop3 сервисы, и т.д.) должны быть совместимы с Shadow. То есть, они должны иметь возможность работать с shadowed паролями.

Для того, чтобы включить поддержку shadowed паролей, выполните следующую команду:

```
pwconv
```

Для того, чтобы включить поддержку shadowed паролей групп, выполните следующую команду:

```
grpconv
```

Конфигурация Shadow по умолчанию для программы **useradd** имеет ряд предостережений, и нуждается в дополнительных разъяснениях. Перовое действие для программы **useradd** - создание пользователя и группы с тем же наименованием, как имя пользователя. По умолчанию, идентификатор пользователя (UID) и идентификатор группы (GID) начинаются с 1000. Это означает, что если к программе **useradd** не было добавлено аргументов, каждый пользователь станет членом уникальной группы в системе. Если такое поведение нежелательно, то можно передать аргумент *`-g`* к вызову программы **useradd**. Параметр по умолчанию хранится в файле `/etc/default/useradd`. Вы можете модифицировать два аргумента в этом файле, так как вы считаете необходимым.

`/etc/default/useradd`Объяснения значений параметров:

`GROUP=1000`
Значение аргумента устанавливает начало нумерации, используемой в файле /etc/group. Значение можно модифицировать на ваше усмотрение. Однако обратите внимание, что команда **useradd** никогда не будет повторно использовать UID или GID. Если номер,

указанный в значении аргумента будет занят, будет использован следующий номер по порядку. Если в системе нет номера 1000 в первый момент использования команды **useradd** без аргумента *-g* , вы получите сообщение, в котором будет следующая информация: `useradd: unknown GID 1000`. Можно игнорировать сообщение, и номер группы 1000 будет использован.

CREATE_MAIL_SPOOL=yes

Аргумент сообщает команде **useradd** создать файл почтового ящика для нового пользователя. **useradd** назначит этому файлу группу `mail` и права 0660. Если вам не нужно, чтобы создавались эти файлы, выполните следующую команду:

```
sed -i 's/yes/no/' /etc/default/useradd
```

6.20.3. Установка пароля для корневного пользователя (root)

Придумайте пароль для пользователя *root* и укажите его, выполнив команду:

```
passwd root
```

6.20.4. Содержимое пакета Shadow

Установленные программы: chage, chfn, chgpasswd, chpasswd, chsh, expiry, faillog, gpasswd, groupadd, groupdel, groupmems, groupmod, grpck, grpconv, grpunconv, lastlog, login, logoutd, newgidmap, newgrp, newuidmap, newusers, nologin, passwd, pwck, pwconv, pwunconv, sg (link to newgrp), su, useradd, userdel, usermod, vigr (link to vipw), and vipw

Каталог установки: /etc/default

Краткое описание

chage Изменение максимального количества дней между обязательными сменами пароля

chfn Изменение полного имени пользователя и другой информации

chgpasswd Обновление групповых паролей в пакетном режиме

chpasswd Обновление пользовательских паролей в пакетном режиме

chsh Используется для изменения оболочки пользователя по умолчанию

expiry Проверяет и применяет текущую политику истечения срока действия пароля

faillog Используется для проверки журнала ошибок входа, для установки максимального значения количество сбоев до блокировки учетной записи или сброс количества неудачных попыток

gpasswd Используется для добавления и удаления членов и администраторов группы

groupadd Создает группу с заданным именем

groupdel Удаляет группу с заданным именем

groupmems Позволяет пользователю управлять своим списком групп без требования привилегий корневого пользователя (root).

groupmod Используется для изменения наименования указанной группы или идентификтора группы (GID)

grpck Проверяет целостность файлов группы `/etc/group` и `/etc/gshadow`

grpconv Создает или обновляет shadow файл группы из обычного файла группы

grpunconv Обновляет файл `/etc/group` из `/etc/gshadow` и затем удаляет последний

lastlog Сообщает о последнем входе в систему для всех пользователей или указанного пользователя

login Используется для входа пользователей в систему

logoutd Используется ли сервис для ограничения времени входа в систему и порты

newgidmap Используется для установки отображения идентификатора группы (GID) пространства имен пользователя

newgrp Используется для изменения текущего идентификатора группы (GID) во время сеанса входа в систему

newuidmap Используется для установки отображения идентификатора пользователя (UID) пространства имен пользователя

newusers Используется для создания или обновления всей серии пользователей

nologin Отображает сообщение о недоступности учетной записи; создан для использования в качестве оболочки по умолчанию для учетных записей, которые были отключены

passwd Используется для изменения пароля для учетной записи пользователя или группы

pwck Проверяет целостность файлов паролей `/etc/passwd` and `/etc/shadow`

pwconv Создает или обновляет shadow файл пароля из обычного файла паролей

pwunconv Обновляет `/etc/passwd` из `/etc/shadow` и затем, удаляет последний

sg Выполняет заданную команду, пока пользовательский GID устанавливается в соответствии с заданной группой

su Запускает оболочку с заменённым идентификатором пользователя и группы

useradd Создает нового пользователя с заданным именем или обновляет значение по умолчанию информации для нового пользователя

userdel Удаляет указанного пользователя

usermod Используется для изменения имени пользователя, пользователя Идентификация (UID), оболочка, начальная группа, домашний каталог и т.д.

vigr Редактирует файлы `/etc/group` или `/etc/gshadow`

vipw Редактирует файлы `/etc/passwd` или `/etc/shadow` files

6.21. GCC-9.2.0

Пакет содержит набор компиляторов GNU, для таких языков как Си и Си++.

Приблизительное время сборки:	95 SBU (with tests)
Требуемое дисковое пространство:	4.2 GB

6.21.1. Установка пакета GCC

Если сборка пакета будет происходить на 64-битной машине, то необходимо сменить каталог. По умолчанию для 64-битных библиотек на каталог “lib”:

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

В документации к пакету рекомендуется выполнять процедуру сборки из отдельного каталога:

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

Подготовьте пакет GCC к компиляции:

```
SED=sed                               \
../configure --prefix=/usr            \
             --enable-languages=c,c++ \
             --disable-multilib       \
             --disable-bootstrap      \
             --with-system-zlib
```

Обратите внимание, что другие языки имеют некоторые дополнительные зависимости, которые на данный момент еще не доступны. Обратитесь к документации *книги BLFS* чтобы узнать как выполнить сборку GCC со всеми поддерживаемыми языками программирования.

Значения новых параметров конфигурации:

`SED=sed`
: Установка этой переменной среды предотвращает использование жестко закодированного пути к каталогу /tools/bin/sed.

`--with-system-zlib`
: Аргумент сообщает, чтобы GCC использовал ссылку на установленную ранее в системе библиотеку Zlib, вместо использования встроенной версии.

Скомпилируйте пакет:

```
make
```

Important

Выполнение тестов пакета GCC считается критичным. Не пропускайте их выполнение ни при каких обстоятельствах.

Известно, что один набор тестов GCC исчерпывает стек, поэтому увеличьте его размер до запуска тестов:

```
ulimit -s 32768
```

Проверьте результаты тестирования от непривилегированного пользователя, но не останавливайтесь на ошибках:

```
chown -Rv nobody .
su nobody -s /bin/bash -c "PATH=$PATH make -k check"
```

Чтобы получить сводку результатов о выполнении тестов, запустите:

```
../contrib/test_summary
```

Чтобы увидеть только сводку **grep -A7 Summ**.

Ваши результаты можно сравнить с результатами, расположенными по ссылке *https://linuxfromscratch.ru/lfs/build-logs/9.0/* и *https://gcc.gnu.org/ml/gcc-testresults/*.

Шесь тестов, которые относятся к get_time известно, что не пройдут. Это явно связано с en_HK locale.

Два теста - lookup.cc и reverse.cc в experimental/net известно что не пройдут в chroot среде LFS, потому что требуют /etc/hosts and iana-etc.

Два теста - pr57193.c и pr90178.c известно что не пройдут.

Несколько неожиданных сбоев не всегда удается избежать. Разработчики GCC как правило знают о таких проблемах, но некоторые из них могут быть ещё не решены. В частности известно, что шесть тестов в библиотеке libstdc++ не пройдут, если выполнять тестирование от имени пользователя root, как мы и выполняем. Если результаты выполнения тестов не особо отличаются от результатов, опубликованных по ссылку выше, то можно продолжать далее.

Установите пакет и удалите ненужный каталог:

```
make install
rm -rf /usr/lib/gcc/$(gcc -dumpmachine)/9.2.0/include-fixed/bits/
```

The GCC build directory is owned by nobody now and the ownership of the installed header directory (and its content) will be incorrect. Change the ownership to `root` user and group:

```
chown -v -R root:root \
    /usr/lib/gcc/*linux-gnu/9.2.0/include{,-fixed}
```

По историческим причинам, создайте символическую ссылку и*FHS*

```
ln -sv ../usr/bin/cpp /lib
```

Многие пакеты используют команду **cc** для вызова компилятора языка Си. Для соблюдения совместимости, добавим следующую символическую ссылку:

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

Добавим ещё одну символическую ссылку, для соблюдения совместимости и возможности сборки программ с поддержкой LTO (Оптимизация времени линковки):

```
install -v -dm755 /usr/lib/bfd-plugins
ln -sfv ../../libexec/gcc/$(gcc -dumpmachine)/9.2.0/liblto_plugin.so \
        /usr/lib/bfd-plugins/
```

Теперь, когда все инструменты на своем месте, важно снова убедиться в том, что всё компиляция и линковка работают правильно. Выполним проверки, аналогичную тем, что выполняли на предыдущих этапах:

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

В результате выполнения этой команды не должно быть ошибок, и вывод (в зависимости от платформы) должен соответствовать следующей строке:

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

Теперь убедимся что настройка стартовых файлов запуска выполнена правильно:

```
grep -o '/usr/lib.*/crt[1in].*succeeded' dummy.log
```

Вывод должен соответствовать следующей строке:

```
/usr/lib/gcc/x86_64-pc-linux-gnu/9.2.0/../../../../lib/crt1.o succeeded
/usr/lib/gcc/x86_64-pc-linux-gnu/9.2.0/../../../../lib/crti.o succeeded
/usr/lib/gcc/x86_64-pc-linux-gnu/9.2.0/../../../../lib/crtn.o succeeded
```

В зависимости от архитектуры машины, вывод, указанный выше, может немного отличаться. Разница может быть в наименовании каталогов после `/usr/lib/gcc` . Важным моментом на который стоит обратить внимание, это наличие трёх файлов `crt*.o` которые ищет **gcc** в каталоге `/usr/lib` .

Проверьте, что компилятор выполняет поиск заголовочных файлов в нужных местах:

```
grep -B4 '^ /usr/include' dummy.log
```

Вывод должен соответствовать следующей строке:

```
#include <...> search starts here:
 /usr/lib/gcc/x86_64-pc-linux-gnu/9.2.0/include
 /usr/local/include
 /usr/lib/gcc/x86_64-pc-linux-gnu/9.2.0/include-fixed
 /usr/include
```

Снова обратите внимание, что наименование каталога. после целевого триплета может отличаться, в зависимости от архитектуры машины.

Далее проверим, что компоновщик использует правильные пути поиска:

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

Ссылки на пути с наличием у компонента суффикса '-linux-gnu' должны быть проигнорированы, и вывод должен соответствовать следующей строке:

```
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");
```

На 32-битных машинах можно заместить что появились некоторые другие каталоги. Например, вывод на архитектуре i686 может быть такой:

```
SEARCH_DIR("/usr/i686-pc-linux-gnu/lib32")
SEARCH_DIR("/usr/local/lib32")
SEARCH_DIR("/lib32")
SEARCH_DIR("/usr/lib32")
SEARCH_DIR("/usr/i686-pc-linux-gnu/lib")
SEARCH_DIR("/usr/local/lib")
SEARCH_DIR("/lib")
SEARCH_DIR("/usr/lib");
```

Проверим что используется нужная библиотека libc:

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

Вывод должен соответствовать следующей строке:

```
attempt to open /lib/libc.so.6 succeeded
```

Наконец, проверим что GCC использует правильный динамический компоновщик:

```
grep found dummy.log
```

Вывод (в зависимости от платформы в наименовании динамического компоновщика) должен соответствовать следующей строке:

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

Если вывод на выполнение команд не выводиться или команды не выполняются вовсе, то это признак того, что в предыдущих инструкциях была допущена серьезная ошибка. Исследуйте предыдущие шаги и проверьте правильность их выполнения. Постарайтесь найти ошибку в вводимых командах. Наиболее вероятно, что причина в проблемах - не правильная настройка специальных файлов при перенастройке временного набора инструментов. Все проблемы должны быть найдены и исправлены здесь, до выполнения следующих этапов.

Если все работает правильно, выполним очистку тестовых файлов:

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

Наконец переместим файлы в правильное местоположение:

```
mkdir -pv /usr/share/gdb/auto-load/usr/lib
mv -v /usr/lib/*gdb.py /usr/share/gdb/auto-load/usr/lib
```

6.21.2. Содержимое пакета GCC

Установленные программы:	c++, cc (link to gcc), cpp, g++, gcc, gcc-ar, gcc-nm, gcc-ranlib, gcov, gcov-dump, and gcov-tool
Установленные библиотеки:	libasan.{a,so}, libatomic.{a,so}, libcc1.so, libgcc.a, libgcc_eh.a, libgcc_s.so, libgcov.a, libgomp.{a,so}, libitm.{a,so}, liblsan.{a,so}, liblto_plugin.so, libquadmath.{a,so}, libssp.{a,so}, libssp_nonshared.a, libstdc++.{a,so}, libstdc++fs.a, libsupc++.a, libtsan.{a,so}, and libubsan.{a,so}
Установленные каталоги:	/usr/include/c++, /usr/lib/gcc, /usr/libexec/gcc, and /usr/share/gcc-9.2.0

Краткое описание

c++	Компилятор Си++
cc	Компилятор Си
cpp	Препроцессор Си; используется компилятором для замены в исходном коде инструкций #include, #define и аналогичных
g++	Компилятор Си++
gcc	Компилятор Си
gcc-ar	Обёртка для команды **ar** добавляющая плагин к командной строке. Эта программа используется только чтобы добавлять опцию "Оптимизация времени линковки" и не особо полезна для выполнения сборки с аргументами по умолчанию.
gcc-nm	Обёртка для команды **nm** добавляющая плагин к командной строке. Эта программа используется только чтобы добавлять опцию "Оптимизация времени линковки" и не особо полезна для выполнения сборки с аргументами по умолчанию.
gcc-ranlib	Обёртка для команды **ranlib** добавляющая плагин к командной строке. Эта программа используется только чтобы добавлять опцию "Оптимизация времени линковки" и не особо полезна для выполнения сборки с аргументами по умолчанию.
gcov	Инструмент покрытия тестами; он используется для анализа программ и определяет, будет-ли оптимизация наиболее эффективной
gcov-dump	Offline gcda and gcno profile dump tool
gcov-tool	Offline gcda profile processing tool
`libasan`	Библиотека времени выполнения очистки адресного пространства
`libatomic`	GCC atomic built-in runtime library

`libcc1`	The C preprocessing library
`libgcc`	Содержит поддержка времени выполнения для **gcc**
`libgcov`	Библиотека связывается с указанной программой. когда GCC необходимо выполнить профилирование
`libgomp`	Реализация проекта GNU API библиотеки OpenMP для кроссплатформенного программирования в разделяемой памяти парралельного программирования на языках Си,Си ++ и Фортран
`liblsan`	The Leak Sanitizer runtime library
`liblto_plugin`	GCC's расширение оптимизации времени линковки (LTO), GCC позволяющее выполнять оптимизацию при компиляции модулей
`libquadmath`	Библиотека API для выполнения математических вычислений четырехмерной точности
`libssp`	Содержит программы для обеспечения защиты стека от атак переполнения буфера в стеке (stack-smashing).
`libstdc++`	Стандартная библиотека Си++
`libstdc++fs`	ISO/IEC TS 18822:2015 Filesystem library
`libsupc++`	Предоставляет вспомогательные процедуры для языка программирования Си ++
`libtsan`	Библиотека времени выполнения очистки потоков
`libubsan`	The Undefined Behavior Sanitizer runtime library

6.22. Bzip2-1.0.8

Пакет содержит программы для сжатия и распаковки файлов. Он необходим для распаковки многих пакетов LFS. Сжатие текстовых файлов при помощи программы **bzip2** даёт больший процент сжатия чем **gzip**.

Приблизительное время сборки: less than 0.1 SBU
Требуемое дисковое пространство: 7.7 MB

6.22.1. Установка пакета Bzip2

Примените патч (исправление) который позволит установить документацию к пакету:

```
patch -Np1 -i ../bzip2-1.0.8-install_docs-1.patch
```

Следующая команда гарантирует установку символических ссылок по относительному пути:

```
sed -i 's@\(ln -s -f \)$(PREFIX)/bin/@\1@' Makefile
```

Убедиться что страницы руководств будут установлены в правильное место:

```
sed -i "s@(PREFIX)/man@(PREFIX)/share/man@g" Makefile
```

Подготовьте пакет Bzip2 для компиляции:

```
make -f Makefile-libbz2_so
make clean
```

Значение параметров make:

`-f Makefile-libbz2_so`
Аргумент сообщает Bzip2 выполнить сборку используя другой `Makefile`, при указании значения аргумента `Makefile-libbz2_so` позволит создать динамическую библиотеку `libbz2.so` и ссылки с Bzip2.

Скомпилируйте и выполните тестирование пакета:

```
make
```

Установите программы:

```
make PREFIX=/usr install
```

Установите общие **bzip2** бинарные файлы в каталог `/bin` , сделайте несколько необходимых символических ссылок и затем, выполните очистку:

```
cp -v bzip2-shared /bin/bzip2
cp -av libbz2.so* /lib
ln -sv ../../lib/libbz2.so.1.0 /usr/lib/libbz2.so
rm -v /usr/bin/{bunzip2,bzcat,bzip2}
ln -sv bzip2 /bin/bunzip2
ln -sv bzip2 /bin/bzcat
```

6.22.2. Содержимое пакета Bzip2

Установленные программы: bunzip2 (link to bzip2), bzcat (link to bzip2), bzcmp (link to bzdiff), bzdiff, bzegrep (link to bzgrep), bzfgrep (link to bzgrep), bzgrep, bzip2, bzip2recover, bzless (link to bzmore), and bzmore

Установленные библиотеки: libbz2.{a,so}

Каталог установки: /usr/share/doc/bzip2-1.0.8

Краткое описание

bunzip2 Распаковывает файлы bzip

bzcat Распаковка в стандартный вывод

bzcmp Запускает программу **cmp** для запакованных файлов

bzdiff Запускает программу **diff** для запакованных файлов

bzegrep Запускает программу **egrep** для запакованных файлов

bzfgrep Запускает программу **fgrep** для запакованных файлов

bzgrep Запускает программу **grep** для запакованных файлов

bzip2 Утилита командной строки для сжатия данных, реализация алгоритма Барроуза — Уилера. Программа bzip2 была написана для замены программы **gzip**. Она действительно показывает лучшие характеристики по степени сжатия данных, но, с другой стороны она потребляет значительно больше памяти при работе

bzip2recover Выполняет попытку восстановить данные из поврежденного файла архива

bzless Запускает программу **less** для запакованных файлов

bzmore Запускает программу **more**для запакованных файлов

`libbz2` Библиотека, реализующая блочную сортировку данных без потерь сжатие с использованием алгоритма Барроуза — Уилера

6.23. Pkg-config-0.29.2

Утилита, предоставляющая интерфейс для получения информации об установленных программных библиотеках, включающую в себя параметры для Си или Си++ компилятора, параметры для компоновщика, а также версию пакета. Инструмент позволяет передавать для включения пути или пути к библиотекам инструментам для сборки во время процесса конфигурирования (configure) и выполнения команд программы make.

Приблизительное время сборки: 0.4 SBU
Требуемое дисковое пространство: 30 MB

6.23.1. Установка пакета Pkg-config

Подготовьте пакет Pkg-config к компиляции:

```
./configure --prefix=/usr              \
            --with-internal-glib       \
            --disable-host-tool        \
            --docdir=/usr/share/doc/pkg-config-0.29.2
```

Значения новых параметров конфигурации:

`--with-internal-glib`
Аргумент позволяет использовать внутреннюю версию пакета Glib, потому, как внешняя версия не доступна в книге LFS.

`--disable-host-tool`
Этот параметр отключает создание нежелательной жесткой ссылки на программу pkg-config.

Скомпилируйте пакет:

```
make
```

Для выполнения тестов, выполните команду:

```
make check
```

Установите пакет:

```
make install
```

6.23.2. Содержимое пакета Pkg-config

Установленная программа: pkg-config
Каталог установки: /usr/share/doc/pkg-config-0.29.2

Краткое описание

pkg-config Возвращает метаданные для указанной библиотеки или пакета

6.24. Ncurses-6.1

Пакет содержит библиотеку, предназначенную для управления вводом-выводом на терминал, в числе прочего, библиотека позволяет задавать экранные координаты (в знакоместах) и цвет выводимых символов. Предоставляет программисту уровень абстракции, позволяющий не беспокоиться об аппаратных различиях терминалов и писать переносимый код. Он необходим для ряда пакетов.

Приблизительное время сборки: 0.4 SBU
Требуемое дисковое пространство: 42 MB

6.24.1. Установка пакета Ncurses

Не следует устанавливать статическую библиотеку, которая не обрабатывается сценарием configure:

```
sed -i '/LIBTOOL_INSTALL/d' c++/Makefile.in
```

Подготовьте пакет Ncurses к компиляции:

```
./configure --prefix=/usr           \
            --mandir=/usr/share/man \
            --with-shared           \
            --without-debug         \
            --without-normal        \
            --enable-pc-files       \
            --enable-widec
```

Значения новых параметров конфигурации:

`--enable-widec`
Этот аргумент указывает, что необходимо скомпилировать библиотеки расширенных символов (таких как, `libncursesw.so.6.1`) вместо обычных (таких как, `libncurses.so.6.1`). Эти библиотеки расширенных символов используются и в многобайтовой и традиционной 8-битной локали, в то время как обычные библиотеки работают должным образом только в 8-битных локалях. библиотеки расширенных символов и обычные совместимы на уровне исходного кода, но не совместимы в двоичном.

`--enable-pc-files`
Этот аргумент создает и устанавливает .pc файлы для pkg-config.

`--without-normal`
Этот аргумент отключает сборку и установку большинства статических библиотек.

Скомпилируйте пакет:

```
make
```

У пакета присутствуют наборы тестов, однако их запуск возможен только после установки пакета. Наборы тестов располагаются в каталоге `test/`. Прочитайте файл `README` в этом каталоге для получения дополнительной информации.

Установите пакет:

```
make install
```

Переместите разделяемые библиотеки в каталог `/lib` где они и должны находиться:

```
mv -v /usr/lib/libncursesw.so.6* /lib
```

Поскольку разделяемые библиотеки были перемещены, одна символическая ссылка теперь указывает на несуществующий файл. Создадим эту ссылку заново:

```
ln -sfv ../../lib/$(readlink /usr/lib/libncursesw.so) /usr/lib/libncursesw.so
```

Многие приложения по-прежнему ожидают, что компоновщик сможет найти библиотеки Ncurses не расширенных символов, а обычных. Обмануть такие программы и связать их с библиотеками расширенных символов при помощи создания символических ссылок и сценариев компоновщика:

```
for lib in ncurses form panel menu ; do
    rm -vf                    /usr/lib/lib${lib}.so
    echo "INPUT(-l${lib}w)" > /usr/lib/lib${lib}.so
    ln -sfv ${lib}w.pc        /usr/lib/pkgconfig/${lib}.pc
done
```

Наконец, убедимся что старые программы будут искать файлы `-lcurses` во время сборки и останутся пригодными для последующей сборки:

```
rm -vf                     /usr/lib/libcursesw.so
echo "INPUT(-lncursesw)" > /usr/lib/libcursesw.so
ln -sfv libncurses.so      /usr/lib/libcurses.so
```

При желании установите документацию к Ncurses:

```
mkdir -v       /usr/share/doc/ncurses-6.1
cp -v -R doc/* /usr/share/doc/ncurses-6.1
```

Note

Приведенные выше инструкции не создают библиотеки расширенных символов, так как ни один пакет, установленный путем компиляции из исходного кода, не будет ссылаться на них во время выполнения. Тем не менее, единственные известные двоичные приложения, которые связываются с библиотеками Ncurses нерасширенных символов, требуют версии 5. Если такие библиотеки необходимы из-за какого-либо приложения только для двоичных файлов или для совместимости с LSB, выполните компиляцию пакета снова с помощью следующих команд:

```
make distclean
./configure --prefix=/usr    \
            --with-shared    \
            --without-normal \
            --without-debug  \
            --without-cxx-binding \
            --with-abi-version=5
make sources libs
cp -av lib/lib*.so.5* /usr/lib
```

6.24.2. Содержимое пакета Ncurses

Установленные программы:	captoinfo (link to tic), clear, infocmp, infotocap (link to tic), ncursesw6-config, reset (link to tset), tabs, tic, toe, tput, and tset
Установленные библиотеки:	libcursesw.so (symlink and linker script to libncursesw.so), libformw.so, libmenuw.so, libncursesw.so, libncurses++w.a, libpanelw.so, and their non-wide-character counterparts without "w" in the library names.
Установленные каталоги:	/usr/share/tabset, /usr/share/terminfo, and /usr/share/doc/ncurses-6.1

Краткое описание

captoinfo	Преобразовывает termcap описание в terminfo
clear	По возможности, очищает экран
infocmp	Сравнивает или распечатывает описания terminfo
infotocap	Преобразует описание terminfo в описание termcap
ncursesw6-config	Предоставляет сведения о конфигурации для ncurses
reset	Повторная инициализация терминала до значений по умолчанию
tabs	Очищает и устанавливает табуляторы на терминале
tic	Компилятор описания записи terminfo, преобразующий файл terminfo из исходного формата в двоичный формат, необходимый для процедур библиотеки ncurses [файл terminfo содержит информацию о возможности определенного терминала.]
toe	Выводит список всех доступных типов терминалов с основным именем и описанием

tput	Принимает значения зависимыз от терминала доступных в оболочке ; ее можно также использовать для того чтобы переустановить или инициализировать терминал или сообщить его полное наименование
tset	Может быть использована для инициализации терминала
`libcursesw`	Ссылка на файл `libncursesw`
`libncursesw`	Содержит функции для отображения текста многими сложными способами на экране терминала; хорошим примером использования этих функций является меню, отображаемое во время создания файла конфигурации ядра **make menuconfig**
`libformw`	Содержит функции для реализации форм
`libmenuw`	Содержитфункции для реализации меню
`libpanelw`	Содержит функции для реализации панелей

6.25. Attr-2.4.48

Программы для администрирования расширенных атрибутов объектов файловой системы.

Приблизительное время сборки: less than 0.1 SBU
Требуемое дисковое пространство: 4.2 MB

6.25.1. Установка пакета Attr

Подготовьте пакет Attr к компиляции:

```
./configure --prefix=/usr     \
            --bindir=/bin     \
            --disable-static  \
            --sysconfdir=/etc \
            --docdir=/usr/share/doc/attr-2.4.48
```

Скомпилируйте пакет:

```
make
```

Тесты необходимо запускать на тех файловых системах, в которых есть поддержка расширенных файловых атрибутов. Например, такая поддержка присутствует в файловых системах ext2, ext3, и ext4. Для выполнения тестов, выполните команду:

```
make check
```

Установите пакет:

```
make install
```

Разделяемые библиотеки необходимо перенести в каталог `/lib` , и в результате файлы `.so` в каталоге `/usr/lib` необходимо создать заново:

```
mv -v /usr/lib/libattr.so.* /lib
ln -sfv ../../lib/$(readlink /usr/lib/libattr.so) /usr/lib/libattr.so
```

6.25.2. Содержимое пакета Attr

Установленные программы: attr, getfattr, and setfattr
Установленная библиотека: libattr.so
Установленные каталоги: /usr/include/attr and /usr/share/doc/attr-2.4.48

Краткое описание

attr Расширяет атрибуты объектов файловой системы

getfattr Получает расширенные атрибуты объектов файловой системы

setfattr Задает дополнительные атрибуты объектов файловой системы

`libattr` Библиотека содержит функции для работы с расширенными атрибутами

6.26. Acl-2.2.53

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 списками управления доступом (ACL).

Приблизительное время сборки: less than 0.1 SBU
Требуемое дисковое пространство: 6.4 MB

6.26.1. Установка пакета Acl

Подготовьте пакет Acl к компиляции:

```
./configure --prefix=/usr         \
            --bindir=/bin         \
            --disable-static      \
            --libexecdir=/usr/lib \
            --docdir=/usr/share/doc/acl-2.2.53
```

Скомпилируйте пакет:

```
make
```

Тесты необходимо запустить на файловой системе, которая поддерживает списки управления доступом, после сборки пакета Coreutils с библиотеками Acl. При желании, вернитесь позднее к этому пакету и выполните команду **make check** после сборки пакета Coreutils который будет установлен немного позднее, в этой главе

Установите пакет:

```
make install
```

Разделяемые библиотеки необходимо перенести в каталог `/lib` , и в результате файлы `.so` в каталоге `/usr/lib` необходимо создать заново:

```
mv -v /usr/lib/libacl.so.* /lib
ln -sfv ../../lib/$(readlink /usr/lib/libacl.so) /usr/lib/libacl.so
```

6.26.2. Содержимое пакета Acl

Установленные программы: chacl, getfacl, and setfacl
Установленная библиотека: libacl.so
Установленные каталоги: /usr/include/acl and /usr/share/doc/acl-2.2.53

Краткое описание

chacl Изменение списка управления доступом к файлу или к каталогу

getfacl Получает списки управления доступом к файлам

setfacl Устанавливает списки управления доступом к файлам

`libacl` Библиотека содержит функции для управления списками управления доступом

6.27. Libcap-2.27

Пакет Libcap реализует интерфейсы пользовательского пространства для возможностей POSIX 1003.1 e, доступных в ядрах Linux. Эти возможности представляют собой разделение привилегий All powerful root на набор различных привилегий.

Приблизительное время сборки: less than 0.1 SBU
Требуемое дисковое пространство: 1.5 MB

6.27.1. Установка пакета Libcap

Запретить установку статической библиотеки

```
sed -i '/install.*STALIBNAME/d' libcap/Makefile
```

Скомпилируйте пакет:

```
make
```

У этого пакета нет тестов.

Установите пакет:

```
make RAISE_SETFCAP=no lib=lib prefix=/usr install
chmod -v 755 /usr/lib/libcap.so.2.27
```

Значение аргументов команды make:

RAISE_SETFCAP=no
Этот параметр пропускает попытки использовать **setcap** для себя. Это позволяет избежать ошибки установки, если ядро или файловая система не поддерживает расширенные возможности.

lib=lib
Аргумент указывает, что установка библиотеки должна происходить в каталог `$prefix/lib` вместо `$prefix/lib64` на архитектурах x86_64. Это не влияет на машины с архитектурой x86.

Разделяемые библиотеки необходимо перенести в каталог `/lib` , и в результате файлы `.so` file in `/usr/lib` необходимо создать заново:

```
mv -v /usr/lib/libcap.so.* /lib
ln -sfv ../../lib/$(readlink /usr/lib/libcap.so) /usr/lib/libcap.so
```

6.27.2. Содержимое пакета Libcap

Установленные программы: capsh, getcap, getpcaps, and setcap
Установленная библиотека: libcap.so

Краткое описание

capsh Обёртка к оболочке для изучения и ограничения поддержки возможностей

getcap Проверяет возможности файлов

getpcaps Отображает возможности запрашиваемого процесса или процессов

setcap Устанавливает возможности файлов

`libcap` Библиотека содержит функции для управления возможностями POSIX 1003.1e

6.28. Sed-4.7

Sed - потоковый текстовый редактор (а также язык программирования), применяющий различные предопределённые текстовые преобразования к последовательному потоку текстовых данных. Он необходим для многих пакетов LFS, на этапе конфигурирования.

Приблизительное время сборки: 0.4 SBU
Требуемое дисковое пространство: 32 MB

6.28.1. Установка пакета Sed

Сначала устраните проблему в среде LFS и удалите ошибочный тест:

```
sed -i 's/usr/tools/'                 build-aux/help2man
sed -i 's/testsuite.panic-tests.sh//' Makefile.in
```

Подготовьте пакет Sed к компиляции:

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

Выполните компиляцию пакета, и сгенерируйте HTML документацию:

```
make
make html
```

Для выполнения тестов, выполните команду:

```
make check
```

Установите пакет и его документацию:

```
make install
install -d -m755           /usr/share/doc/sed-4.7
install -m644 doc/sed.html /usr/share/doc/sed-4.7
```

6.28.2. Содержимое пакета Sed

Установленная программа: sed
Каталог установки: /usr/share/doc/sed-4.7

Краткое описание

sed Фильтрация и преобразование текстовых файлов за один проход

6.29. Psmisc-23.2

Пакет Psmisc содержит программы для отображения информации о запущенных процессах.

Приблизительное время сборки: less than 0.1 SBU
Требуемое дисковое пространство: 4.6 MB

6.29.1. Установка пакета Psmisc

Подготовьте пакет Psmisc к компиляции:

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

Скомпилируйте пакет:

```
make
```

У этого пакета нет тестов.

Установите пакет:

```
make install
```

Переместите файлы программ **killall** и **fuser** в место, определённое стандартом FHS:

```
mv -v /usr/bin/fuser   /bin
mv -v /usr/bin/killall /bin
```

6.29.2. Содержимое пакета Psmisc

Установленные программы: fuser, killall, peekfd, prtstat, pslog, pstree, and pstree.x11 (ссылка на pstree)

Краткое описание

fuser Сообщает идентификатор процесса (PID) процесса, который использует данный файл или файловая система

killall Уничтожает процесс по наименованию; посылает сигнал всем процессам, которые были запущены указанной командой.

peekfd Просмотр файловых дескрипторов запущенного процесса, учитывая его PID

prtstat Отображает информацию о процессе

pslog Reports current logs path of a process

pstree Отображает запущенные процессы в виде дерева

pstree.x11 Тоже самое что **pstree**, кроме того, что команда будет ожидать подтверждения перед выходом

6.30. Iana-Etc-2.30

Пакет Iana-Etc содержит данные для сетевых служб и протоколов.

Приблизительное время сборки: less than 0.1 SBU
Требуемое дисковое пространство: 2.3 MB

6.30.1. Установка пакета Iana-Etc

Следующая команда преобразует необработанные данные, предоставленные IANA, в требуемые форматы для файлов с данными `/etc/protocols` и `/etc/services`

```
make
```

У этого пакета нет тестов.

Установите пакет:

```
make install
```

6.30.2. Содержимое пакета Iana-Etc

Установленные файлы: /etc/protocols and /etc/services

Краткое описание

`/etc/protocols` Описывает различные интернет-протоколы DARPA, которые доступны из подсистемы TCP/IP

`/etc/services` Обеспечивает соответствие между текстовыми именами для интернет служб и назначенными номерами портов и протоколов

6.31. Bison-3.4.1

Пакет содержит GNU версию yacc (Ещё один компилятор компиляторов) необходимый для сборки некоторых пакетов LFS системы.

Приблизительное время сборки: 0.3 SBU
Требуемое дисковое пространство: 39 MB

6.31.1. Установка пакета Bison

First, fix a build problem with the current version:

```
sed -i '6855 s/mv/cp/' Makefile.in
```

Подготовьте пакет Bison к компиляции:

```
./configure --prefix=/usr --docdir=/usr/share/doc/bison-3.4.1
```

Compile the package, but work around a race condition in the current version:

```
make -j1
```

Есть циклическая зависимость между bison и flex при проверке. При желании, после установки пакета flex в следующем разделе, пакет bison можно пересобрать и наборы тестов можно будет запустить снова командой **make check**.

Установите пакет:

```
make install
```

6.31.2. Содержимое пакета Bison

Установленные программы: bison and yacc
Установленная библиотека: liby.a
Каталог установки: /usr/share/bison

Краткое описание

bison Генерирует из набора правил программу для анализа структуры текстовых файлов. Bison является заменой Yacc (еще один компилятор компиляторов)

yacc Обертка для**bison**,предназначенная для программ, которые все еще вызывают **yacc** вместо **bison**; она вызывает **bison**с аргументом *-y*

`liby` Библиотека Yacc содержит реализацию Yacc-совместимых функций `yyerror` и `main`; Эта библиотека обычно не очень полезна, но POSIX требует её

6.32. Flex-2.6.4

Flex (Fast Lexical Analyzer) — генератор лексических анализаторов. Это инструмент для лексического анализа, который может использоваться для выделения из исходного текста определенных строк заранее заданным способом.

Приблизительное время сборки: 0.5 SBU
Требуемое дисковое пространство: 36 MB

6.32.1. Установка пакета Flex

Сначала, необходимо исправить проблему, обнаруженную в glibc-2.26:

```
sed -i "/math.h/a #include <malloc.h>" src/flexdef.h
```

В процессе процедуры сборки предполагается наличие программы help2man - программа для создания справочной страницы из исполняемого файла --help. Её нет, поэтому мы используем переменную среды, чтобы пропустить этот процесс: Выполните подготовку пакета к компиляции:

```
HELP2MAN=/tools/bin/true \
./configure --prefix=/usr --docdir=/usr/share/doc/flex-2.6.4
```

Скомпилируйте пакет:

```
make
```

Для выполнения тестов (около 0.5 SBU), выполните следующую команду:

```
make check
```

Установите пакет:

```
make install
```

Некоторые программы ничего пока не знают о **flex**, и будут пытаться запустить предшественника **lex**. Чтобы обуспечить возможность использования flex, создадим символическую ссылку с названием **lex** которая запустить требуемую программу `flex` в режиме эмуляции **lex**:

```
ln -sv flex /usr/bin/lex
```

6.32.2. Содержимое пакета Flex

Установленные программы: flex, flex++ (link to flex), and lex (link to flex)
Установленные библиотеки: libfl.so
Каталог установки: /usr/share/doc/flex-2.6.4

Краткое описание

flex Инструмент для генерации программ, распознающих шаблоны в тексте; она позволяет универсально определять правила для поиска паттернов, исключая необходимость разработки специализированной программы

flex++ Расширение для flex, используемая для генерации кода Си++ и классов. Является символической ссылкой на **flex**

lex Символическая ссылка, которая запускает **flex** в режиме эмуляции **lex**

`libfl` Библиотека `flex`

6.33. Grep-3.3

Пакет содержит программу, которая находит на вводе строки, отвечающие заданному регулярному выражению, и выводит их, если вывод не отменён специальным ключом. Пакет используется в процедурах сборки для большинства пакетов.

Приблизительное время сборки: 0.5 SBU
Требуемое дисковое пространство: 37 MB

6.33.1. Установка пакета Grep

Подготовьте пакет Grep к компиляции:

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

Скомпилируйте пакет:

```
make
```

Для выполнения тестов, выполните команду:

```
make -k check
```

Установите пакет:

```
make install
```

6.33.2. Содержимое пакета Grep

Установленные программы: egrep, fgrep, and grep

Краткое описание

egrep Печать строк, соответствующих расширенному регулярному выражению
fgrep Печать строк, соответствующих списку фиксированных строк
grep Печать строк, соответствующих базовому регулярному выражению

6.34. Bash-5.0

Усовершенствованная и модернизированная вариация командной оболочки Bourne shell. Этот пакет выполняет требования стандарта LFS Core для обеспечения интерфейса Bourne Shell в системе. Он был выбран из числа других оболочек из-за широкого распространения, возможностей которые выходят далеко за пределы базовых функций программ-оболочек.

Приблизительное время сборки: 2.1 SBU
Требуемое дисковое пространство: 62 MB

6.34.1. Установка пакета Bash

Подготовьте пакет Bash к компиляции:

```
./configure --prefix=/usr                      \
            --docdir=/usr/share/doc/bash-5.0 \
            --without-bash-malloc              \
            --with-installed-readline
```

Значение новых параметров конфигурации:

`--with-installed-readline`
Аргумент сообщает Bash использовать библиотеку `readline`, которая используется в хост-системе, вместо того, чтобы использовать собственную версию.

Скомпилируйте пакет:

```
make
```

Пропустите этот параграф и перейдите к параграфу “Установка пакета” если нет необходимости запускать наборы тестов.

Подготовьте тесты к выполнению, и обудитесь, что пользователь nobody может имет права на запись в дерево исходных текстов пакета:

```
chown -Rv nobody .
```

Запустите тесты от пользователя nobody:

```
su nobody -s /bin/bash -c "PATH=$PATH HOME=/home make tests"
```

Установите пакет и переместите основные исполняемые файлы в каталог `/bin` :

```
make install
mv -vf /usr/bin/bash /bin
```

Запустите новый скомпилированный **bash** (заменив тот, который в настоящее время выполняется):

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

Note

Аргументы используемые для **bash** сообщают обрабатывать интерактивную оболочку входа и отключить хэширование чтобы новые программы были найдены по мере их появления.

6.34.2. Содержимое пакета Bash

Установленные программы: bash, bashbug, and sh (ссылка на bash)
Каталог установки: /usr/include/bash, /usr/lib/bash, и /usr/share/doc/bash-5.0

Краткое описание

bash Широко используемый командный интерпретатор; он выполняет множество типов расширений и подстановок в заданной командной строке перед выполнением это, что делает интерпретатор мощным инструментом

bashbug Сценарий оболочки, который поможет пользователю составить и отправить по почте стандартное отформатированное сообщение об ошибках, относительно **bash**

sh Сивольнач ссылка на программу **bash**; при вызове команды **sh**, **bash** пытается имитировать начальное поведение исторических версий **sh** как как можно ближе, в соответствии со стандартом POSIX

6.35. Libtool-2.4.6

GNU libtool является общей библиотекой поддержки скриптов. Libtool скрывает сложность использования распределенных библиотек под последовательным, переносимым интерфейсом. Библиотека необходима для выполнения тестов других пакетов LFS.

Приблизительное время сборки: 1.9 SBU

Требуемое дисковое пространство: 43 MB

6.35.1. Установка пакета Libtool

Подготовьте пакет Libtool к компиляции:

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

Скомпилируйте пакет:

```
make
```

Для выполнения тестов, выполните следующую команду:

```
make check
```

Note

Время выполнения тестов можно сократить на машине с несколькими ядрами. Чтобы это сделать добавьте к команде аргумент к вышеуказанной команде: **TESTSUITEFLAGS=-j<N>**. Используя значение -j4 может сократить время выполнения тестов более чем на 60 процентов.

Как известно, пять тестов не пройдут в окружении сборки LFS из-за циклических зависимостей, но все тесты пройдут, если их повторно запустить после установки пакета automake.

Установите пакет:

```
make install
```

6.35.2. Содержимое пакета Libtool

Установленные программы:	libtool and libtoolize
Установленные библиотеки:	libltdl.so
Установленные каталоги:	/usr/include/libltdl and /usr/share/libtool

Краткое описание

libtool	Обеспечивает вспомогательные общие службы для сборки
libtoolize	Обеспечивает стандартный путь для добавления поддержки **libtool** для пакета
`libltdl`	Скрывает различные трудности dlopening библиотек

6.36. GDBM-1.18.1

GDBM - библиотека функций базы данных, которая использует расширяемое хэширование и работает аналогично стандартным функциям dbm UNIX. Эти процедуры предоставляются программисту, которому необходимо создать и обработать хешированную базу данных. Основное предназначение gdbm — хранить пары ключ/данные в файле данных. Каждый ключ должен быть уникальным и сопряженным только с одним элементом данных. Ключи не могут быть напрямую доступны в отсортированном порядке.

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

6.36.1. Установка пакета GDBM

Подготовьте пакет GDBM к компиляции:

```
./configure --prefix=/usr    \
            --disable-static \
            --enable-libgdbm-compat
```

Значения параметров конфигурации:

`--enable-libgdbm-compat`
Аргумент включает возможность совместимости библиотеки libgdbm для сборки, поскольку некоторые пакеты за пределами обсуждений книги, могут требовать более старые механизмы DBM.

Скомпилируйте пакет:

```
make
```

Для выполнения тестов, выполните команду:

```
make check
```

Установите пакет:

```
make install
```

6.36.2. Содержимое пакета GDBM

Установленные программы: gdbm_dump, gdbm_load, and gdbmtool
Установленные библиотеки: libgdbm.so and libgdbm_compat.so

Краткое описание

gdbm_dump Создаёт дамп базы данных GDBM в файл
gdbm_load Восстанавливает базу данных GDBM из файла дампа

gdbmtool	Проверяет и изменяет базу данных GDBM
`libgdbm`	Содержит функции для управления хешированной базой данных
`libgdbm_compat`	Библиотека совместимости, содержащая более старые функции DBM

6.37. Gperf-3.1

Gperf генерирует превосходную хэш-функцию из набора ключей.

Приблизительное время сборки: less than 0.1 SBU
Требуемое дисковое пространство: 6.3 MB

6.37.1. Установка пакета Gperf

Подготовьте пакет Gperf к компиляции:

```
./configure --prefix=/usr --docdir=/usr/share/doc/gperf-3.1
```

Скомпилируйте пакет:

```
make
```

Известно что тесты могут не проходить, если выполнение происходит в парралельном режиме (если значение аргумента -j больше чем 1) Для выполнения тестов, выполните следующую команду:

```
make -j1 check
```

Установите пакет:

```
make install
```

6.37.2. Содержимое пакета Gperf

Установленная программа: gperf
Каталог установки: /usr/share/doc/gperf-3.1

Краткое описание

gperf Генерирует превосходную хэш-функцию из набора ключей

6.38. Expat-2.2.7

Expat потокоориентированная библиотека парсинга XML, написанная на C. Как один из наиболее доступных XML парсеров, широко используется в открытом программном обеспечении.

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

6.38.1. Установка пакета Expat

Исправим проблему при работе с регрессионными тестами в окружении LFS:

```
sed -i 's|usr/bin/env |bin/|' run.sh.in
```

Подготовьте пакет Expat к компиляции:

```
./configure --prefix=/usr    \
            --disable-static \
            --docdir=/usr/share/doc/expat-2.2.7
```

Скомпилируйте пакет:

```
make
```

Для выполнения тестов, выполните команду:

```
make check
```

Установите пакет:

```
make install
```

При желании, можно установить документацию:

```
install -v -dm755 /usr/share/doc/expat-2.2.7
install -v -m644 doc/*.{html,png,css} /usr/share/doc/expat-2.2.7
```

6.38.2. Содержимое пакета Expat

Установленная программа: xmlwf
Установленные библиотеки: libexpat.so
Каталог установки: /usr/share/doc/expat-2.2.7

Краткое описание

xmlwf Утилита для проверки XML документов на предмет правильного оформления

`libexpat` Содержит функции API для разбора (парсинга) XML

6.39. Inetutils-1.9.4

Пакет Inetutils содержит базовые программы для работы с сетью.

Приблизительное время сборки: 0.3 SBU
Требуемое дисковое пространство: 29 MB

6.39.1. Установка пакета Inetutils

Подготовьте пакет Inetutils к компиляции:

```
./configure --prefix=/usr        \
            --localstatedir=/var \
            --disable-logger     \
            --disable-whois      \
            --disable-rcp        \
            --disable-rexec      \
            --disable-rlogin     \
            --disable-rsh        \
            --disable-servers
```

Значение параметров конфигурации:

`--disable-logger`
Аргумент запрещает устанавливать программу **logger**, которая используется сценариями для перенаправления сообщений системной службе логирования (System Log Daemon). Не устанавливайте его, потому что Util-linux устанавливает более новую версию.

`--disable-whois`
Аргумент запрещает устанавливать программу **whois**, которая является устаревшей. Инструкции для установки лучшей альтернативы **whois** располагается в книге BLFS.

`--disable-r*`
Аргумент запрещает устанавливать устаревшие программы которые по соображениям безопасности не должны быть установлены. Функции, поддерживаемые этими программами можно заменить функционалом из пакета openssh из книги BLFS.

`--disable-servers`
Отключает установку различных сетевых серверов из пакета Inetutils. Эти серверы считаются не соответствующих базовой системе LFS. Некоторые из них небезопасны по своей природе и считаются безопасным только в надежных сетях. Обратите внимание, что для многих из этих серверов есть лучшие замены.

Скомпилируйте пакет:

```
make
```

Для выполнения тестов, выполните команду:

```
make check
```

Note

Один тест, libls.sh, может не пройти в среде chroot но пройдет корректно, если повторить проверку после окончательной установки системы LFS. Ещё один тест ping-localhost.sh также не пройдёт если хост система не поддерживает возможности ipv6.

Установите пакет:

```
make install
```

Переместите некоторые программы, чтобы они были доступны, если каталог `/usr` недоступен:

```
mv -v /usr/bin/{hostname,ping,ping6,traceroute} /bin
mv -v /usr/bin/ifconfig /sbin
```

6.39.2. Содержимое пакета Inetutils

Установленные программы: dnsdomainname, ftp, ifconfig, hostname, ping, ping6, talk, telnet, tftp, and traceroute

Краткое описание

dnsdomainname Показывает системное DNS имя

ftp Программа для передачи файлов по протоколу FTP

hostname Указывает, или задаёт имя хоста

ifconfig Управление сетевыми интерфейсами

ping Отправляет пакеты эхо-запросов и сообщает, как долго ответы принимались

ping6 Версия **ping** для IPv6 сетей

talk Используется для общения с другими пользователями

telnet Интерфейс для протокола TELNET

tftp Программа для передачи файлов по протоколу TFTP (Trivial File Transfer Protocol — простой протокол передачи файлов)

traceroute Служебная компьютерная программа, предназначенная для определения маршрутов следования данных в сетях TCP/IP. Traceroute может использовать разные протоколы передачи данных в зависимости от операционной системы. Такими протоколами могут быть UDP, TCP, ICMP или GRE.

6.40. Perl-5.30.0

Высокоуровневый интерпретируемый динамический язык программирования общего назначения, он необходим для установки и выполнения тестов некоторых пакетов LFS.

Приблизительное время сборки: 9.9 SBU
Требуемое дисковое пространство: 272 MB

6.40.1. Установка пакета Perl

Для начала создадим файл `/etc/hosts` для определения ссылки в файлах конфигурации Perl и для прохождения опциональных наборов тестов:

```
echo "127.0.0.1 localhost $(hostname)" > /etc/hosts
```

Эта версия Perl теперь собирает Compress::Raw::Zlib и Compress::Raw::BZip2 модули. По умолчанию, Perl будет использовать внутреннюю копию исходных файлов для сборки. Выполните следующую команду, чтобы использовать уже установленные системные библиотеки:

```
export BUILD_ZLIB=False
export BUILD_BZIP2=0
```

Чтобы иметь полный контроль над настройкой, можно убрать аргумент “-des” из нижеуказанной команды и вручную укажите местоположения для сборки. В качестве альтернативы используйте команду, как показано ниже и используйте настройки по умолчанию, которые автоматически обнаружит Perl:

```
sh Configure -des -Dprefix=/usr                 \
                  -Dvendorprefix=/usr           \
                  -Dman1dir=/usr/share/man/man1 \
                  -Dman3dir=/usr/share/man/man3 \
                  -Dpager="/usr/bin/less -isR"  \
                  -Duseshrplib                  \
                  -Dusethreads
```

Значение параметров конфигурации:

`-Dvendorprefix=/usr`
Гарантирует что **perl** будет знать место, куда выполнять установку модулей

`-Dpager="/usr/bin/less -isR"`
Гарантирует что будет использована программа **`less`** вместо **`more`**.

`-Dman1dir=/usr/share/man/man1 -Dman3dir=/usr/share/man/man3`
Поскольку Groff ещё не установлен, сценарий **Configure** думает что нам не требуется документация. Эти аргументы переопределяют это решение.

`-Duseshrplib`
Сборка общей библиотеки libperl необходимая для некоторых модулей Perl

-Dusethreads
Сборка Perl с поддержкой потоков

Скомпилируйте пакет:

```
make
```

Для выполнения тестов (приблизительное время 11 SBU), выполните следующую команду:

```
make -k test
```

Note

Один тест не пройдет из-за использования последней версии gdbm.

Установите пакет and clean up:

```
make install
unset BUILD_ZLIB BUILD_BZIP2
```

6.40.2. Содержимое пакета Perl

Установленные программы:	corelist, cpan, enc2xs, encguess, h2ph, h2xs, instmodsh, json_pp, libnetcfg, perl, perl5.30.0 (hard link to perl), perlbug, perldoc, perlivp, perlthanks (hard link to perlbug), piconv, pl2pm, pod2html, pod2man, pod2text, pod2usage, podchecker, podselect, prove, ptar, ptardiff, ptargrep, shasum, splain, xsubpp, and zipdetails
Установленные библиотеки:	Many which cannot all be listed here
Каталог установки:	/usr/lib/perl5

Краткое описание

corelist	Интерфейс командной строки для модуля Module::CoreList
cpan	Взаимодействие с Comprehensive Perl Archive Network (CPAN) из командной строки
enc2xs	Создает расширение Perl для модуля Encode из Unicode Character Mappings или файлы кодировки Tcl
encguess	Предполагает тип кодировки одного или нескольких файлов
h2ph	Преобразует `.h` заголовочные файлы Си в `.ph` заголовочные файлы Perl
h2xs	Преобразует `.h` заголовочные файлы Си в расширение Perl
instmodsh	Сценарий оболочки для проверки установленных модулей Perl, и создание архива из установленного модуля
json_pp	Преобразует данные между определенными форматами ввода и вывода
libnetcfg	Может использоваться для настройки модуля `libnet`
perl	Объединяет некоторые из лучших возможностей языка Си, **sed**, **awk** и **sh**в один язык
perl5.30.0	Жесткая ссылка на **perl**

perlbug Используется для создания отчетов об ошибках Perl или входящих в состав модулей, для отправки по электронной почте

perldoc Отображает часть документации в формате pod, которая встроена в дерево установки Perl или в сценарии Perl

perlivp Процедура проверки установки Perl; его можно использовать для того чтобы убедиться в корректности установки и настройки Perl и его библиотек

perlthanks Используется для создания благодарственных сообщений на электронную почту разработчиков Perl

piconv Версия Perl конвертера кодировки символов **iconv**

pl2pm Грубый инструмент для преобразования файлов Perl4 `.pl` в модули Perl5 `.pm`

pod2html Преобразует файлы из формата pod в формат HTML

pod2man Преобразует данные pod в форматированны формат * roff

pod2text Преобразует данные pod в форматированный текст ASCII

pod2usage Отображает сообщений об использовании из встроенных документов pod в файлы

podchecker Проверяет синтаксис файлов документации формата pod

podselect Отображение выбранных разделов pod документации

prove Инструмент командной строки для запуска тестов модуля Test :: Harness

ptar Похожая на **tar**программа, написанная на языке Perl

ptardiff Программа Perl, которая сравнивает извлеченный архив с нераспакованной областью

ptargrep Программа Perl, применяющая сопоставление шаблонов к содержимому файлов в архиве tar

shasum Печать или проверка контрольных сумм SHA

splain Используется для диагностики предупреждений в Perl

xsubpp Преобразует Perl XS код в Си код

zipdetails Отображает сведения о внутренней структуре Zip-файла

6.41. XML::Parser-2.44

XML::Parser - Модуль (автор James Clark), который является интерфейсом Perl для библиотеки Expat.

Приблизительное время сборки: less than 0.1 SBU
Требуемое дисковое пространство: 2.3 MB

6.41.1. Установка пакета XML::Parser

Подготовьте пакет XML::Parser к компиляции:

```
perl Makefile.PL
```

Скомпилируйте пакет:

```
make
```

Для выполнения тестов, выполните команду:

```
make test
```

Установите пакет:

```
make install
```

6.41.2. Содержимое пакета XML::Parser

Установленные модули: Expat.so

Краткое описание

`Expat` обеспечивает интерфейс Perl для Expat

6.42. Intltool-0.51.0

Intltool - инструмент интернационализации, используемый для извлечения переводимых строк из исходных файлов.

Приблизительное время сборки: less than 0.1 SBU
Требуемое дисковое пространство: 1.5 MB

6.42.1. Установка пакета Intltool

Сначала исправим предупреждение, вызванное в perl-5.22 и более поздних версиях:

```
sed -i 's:\\\${:\\\$\\{:' intltool-update.in
```

Подготовьте пакет Intltool к компиляции:

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

Скомпилируйте пакет:

```
make
```

Для выполнения тестов, выполните команду:

```
make check
```

Установите пакет:

```
make install
install -v -Dm644 doc/I18N-HOWTO /usr/share/doc/intltool-0.51.0/I18N-HOWTO
```

6.42.2. Содержимое пакета Intltool

Установленные программы: intltool-extract, intltool-merge, intltool-prepare, intltool-update, and intltoolize
Установленные каталоги: /usr/share/doc/intltool-0.51.0 and /usr/share/intltool

Краткое описание

intltoolize Подготовьте пакет для использования intltool
intltool-extract Создает заголовочные файлы, которые могут быть прочитаны программой **gettext**
intltool-merge Объединяет переведенные строки в различные типы файлов
intltool-prepare Обновляет файлы pot и объединяет их с файлами перевода
intltool-update Обновляет файлы шаблонов po и объединяет их с переводами

6.43. Autoconf-2.69

Программы для воспроизведения сценариев командной оболочки которые могут выполнять автоматическую настройку исходного кода из определенного пользовательского файла-шаблона. Они также необходимы для повторной компиляции пакета после обновления процедур сборки.

Приблизительное время сборки: less than 0.1 SBU (about 3.4 SBU with tests)
Требуемое дисковое пространство: 79 MB

6.43.1. Установка пакета Autoconf

Сначала исправьте ошибку, сгенерированную пакетом Perl 5.28.

```
sed '361 s/{/\\{/' -i bin/autoscan.in
```

Подготовьте пакет Autoconf к компиляции:

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

Скомпилируйте пакет:

```
make
```

Для выполнения тестов, выполните команду:

```
make check
```

Установите пакет:

```
make install
```

6.43.2. Содержимое пакета Autoconf

Установленные программы: autoconf, autoheader, autom4te, autoreconf, autoscan, autoupdate, and ifnames
Каталог установки: /usr/share/autoconf

Краткое описание

autoconf Генерирует сценарии оболочки которые автоматически выполняют конфигурирование исходного кода программного обеспечения для адаптации ко многим разновидностям Unix-подобными системам; сценарии конфигурирования, сгенерированные программой autoconf независимы и не требуют программы **autoconf**

autoheader Инструмент для создания шаблона для языка Си из выражения *#define* для конфигурирования

autom4te Обертка для макро процессора M4

autoreconf Автоматически запускает **autoconf**, **autoheader**, **aclocal**, **automake**, **gettextize**, и **libtoolize** в правильном порядке чтобы сохранить время когда были сделаны изменения у файлов-шаблонов **autoconf** и **automake**

autoscan Помогает создать файл `configure.in` для пакета; он проверяет исходные файлы в дереве каталогов, выполняет поиск общих проблем с переносимостью и создаёт файл `configure.scan` который является предварительным файлом `configure.in` пакета

autoupdate Модифицирует файл `configure.in` который по прежнему вызывает макросы **autoconf** по их старым именам для использования имен макросов

ifnames Помогает при написании файла `configure.in` для пакета; команда отображает идентификаторы препроцессора Си которые использует пакет [Если пакет уже настроен для переносимости, эта программа может помочь определить что сценарий **configure** требует для проведения проверки. Он также может заполнить пробелы в файле `configure.in` , сформированным **autoscan**.]

6.44. Automake-1.16.1

Программы для создания файлов Makefile для использования его программой Autoconf. Он также необходим для повторной компиляции пакета после обновления процедур сборки.

Приблизительное время сборки: less than 0.1 SBU (about 8.7 SBU with tests)
Требуемое дисковое пространство: 107 MB

6.44.1. Установка пакета Automake

Подготовьте пакет Automake к компиляции:

```
./configure --prefix=/usr --docdir=/usr/share/doc/automake-1.16.1
```

Скомпилируйте пакет:

```
make
```

Есть несколько тестов, которые ссылаются на неправильную версию библиотеки flex, поэтому мы временно работаем над этой проблемой. Кроме того, использование опции -j4 make ускоряет тесты, даже на системах с одним процессором из-за внутренних задержек в отдельных тестах.Для выполнения тестов, выполните команду:

Using the -j4 make option speeds up the tests, even on systems with only one processor, due to internal delays in individual tests. To test the results, issue:

```
make -j4 check
```

Один тест завершится с ошибкой, при его выполнении в окружении LFS: subobj.sh.

Известно, что два теста не пройдут в среде LFS: check12.sh and check12-w.sh.

Установите пакет:

```
make install
```

6.44.2. Содержимое пакета Automake

Установленные программы: aclocal, aclocal-1.16 (жесткая ссылка на aclocal), automake, и automake-1.16 (жесткая ссылка на automake)
Установленные каталоги: /usr/share/aclocal-1.16, /usr/share/automake-1.16, и /usr/share/doc/automake-1.16.1

Краткое описание

aclocal Формирует файлы `aclocal.m4` на основе содержимого файла `configure.in`

aclocal-1.16 Жесткая ссылка на **aclocal**

automake Инструмент для автоматического создания файлов `Makefile.in` из файлов `Makefile.am` [Чтобы создать файлы `Makefile.in` для пакета, запустите программу в вышестоящем каталоге. Сканируя файл `configure.in` ,

выполняется автоматический поиск каждого подходящего файла `Makefile.am` и формирование соответствующего файла`Makefile.in` .]

automake-1.16 Жесткая ссылка на **automake**

6.45. Xz-5.2.4

Пакет содержит программы для сжатия и распаковки файлов. Он обеспечивает высокое сжатие и используется для распаковки пакетов форматов XZ и LZMA. Сжатие текстовых файлов при помощи программы **xz**даёт больший процент сжатия чем **gzip** или **bzip2**.

Приблизительное время сборки: 0.2 SBU
Требуемое дисковое пространство: 16 MB

6.45.1. Установка пакета Xz

Подготовьте пакет Xz for compilation with:

```
./configure --prefix=/usr    \
            --disable-static \
            --docdir=/usr/share/doc/xz-5.2.4
```

Скомпилируйте пакет:

```
make
```

Для выполнения тестов, выполните команду:

```
make check
```

Установите пакет и убедитесь что основные файлы находятся в необходимом каталоге:

```
make install
mv -v   /usr/bin/{lzma,unlzma,lzcat,xz,unxz,xzcat} /bin
mv -v /usr/lib/liblzma.so.* /lib
ln -svf ../../lib/$(readlink /usr/lib/liblzma.so) /usr/lib/liblzma.so
```

6.45.2. Содержимое пакета Xz

Установленные программы: lzcat (link to xz), lzcmp (link to xzdiff), lzdiff (link to xzdiff), lzegrep (link to xzgrep), lzfgrep (link to xzgrep), lzgrep (link to xzgrep), lzless (link to xzless), lzma (link to xz), lzmadec, lzmainfo, lzmore (link to xzmore), unlzma (link to xz), unxz (link to xz), xz, xzcat (link to xz), xzcmp (link to xzdiff), xzdec, xzdiff, xzegrep (link to xzgrep), xzfgrep (link to xzgrep), xzgrep, xzless, and xzmore
Установленные библиотеки: liblzma.so
Установленные каталоги: /usr/include/lzma and /usr/share/doc/xz-5.2.4

Краткое описание

lzcat Выполняется распаковка в стандартный вывод

lzcmp Запускает программу **cmp** для запакованных файлов при помощи алгоритма LZMA

lzdiff Запускает программу **diff** для запакованных файлов при помощи алгоритма LZMA

lzegrep Запускает программу **egrep** для запакованных файлов при помощи алгоритма LZMA

lzfgrep Запускает программу **fgrep** для запакованных файлов при помощи алгоритма LZMA

lzgrep Запускает программу **grep** для запакованных файлов при помощи алгоритма LZMA

lzless Запускает программу **less** для запакованных файлов при помощи алгоритма LZMA

lzma Запаковывает и распаковывает файлы используя формат LZMA

lzmadec Небольшой и быстрый декодер для запакованных файлов в формате LZMA

lzmainfo Показывает информацию, хранящуюся в заголовке сжатого файла LZMA

lzmore Запускает программу **more** для запакованных файлов при помощи алгоритма LZMA

unlzma Распаковывает файлы используя формат LZMA

unxz Распаковывает файлы используя формат XZ

xz Запаковывает и распаковывает файлы используя формат XZ

xzcat Выполняется распаковка в стандартный вывод

xzcmp Запускает программу **cmp** для сжатых файлов в формте XZ

xzdec Небольшой и быстрый декодер для запакованных файлов в формате XZ

xzdiff Запускает программу **diff** для запакованных файлов в формате XZ

xzegrep Запускает программу **egrep** для запакованных файлов в формате XZ

xzfgrep Запускает программу **fgrep** для запакованных файлов в формате XZ

xzgrep Запускает программу **grep** для запакованных файлов в формате XZ

xzless Запускает программу **less** для запакованных файлов в формате XZ

xzmore Запускает программу **more** для запакованных файлов в формате XZ

`liblzma` LZMA((англ. Lempel-Ziv-Markov chain-Algorithm) Библиотека, реализующая алгоритм на схеме сжатия данных по словарю и обеспечивает высокий коэффициент сжатия (обычно превышающий коэффициент, получаемый при сжатии с использованием bzip2), а также позволяет использовать словари различного размера

6.46. Kmod-26

The Kmod пакет содержит библиотеки и утилиты для загрузки модулей ядра

Приблизительное время сборки: 0.1 SBU
Требуемое дисковое пространство: 13 MB

6.46.1. Установка пакета Kmod

Подготовьте пакет Kmod к компиляции:

```
./configure --prefix=/usr          \
            --bindir=/bin          \
            --sysconfdir=/etc      \
            --with-rootlibdir=/lib \
            --with-xz              \
            --with-zlib
```

Значение параметров конфигурации:

`--with-xz, --with-zlib`
Эти аргументы позволяют Kmod обрабатывать сжатые модули ядра.

`--with-rootlibdir=/lib`
Этот параметр обеспечивает размещение разных файлов, связанных с библиотекой в правильных каталогах.

Скомпилируйте пакет:

```
make
```

В пакете не предусмотрено выполнение тестов в среде LFS. Как минимум требуется программа git и несколько тестов не будут выполняться вне репозитория git.

Установите пакет, и создайте символические ссылки для обеспечения совместимости с Module-Init-Tools (пакет, который ранее обрабатывал модули ядра Linux):

```
make install

for target in depmod insmod lsmod modinfo modprobe rmmod; do
  ln -sfv ../bin/kmod /sbin/$target
done

ln -sfv kmod /bin/lsmod
```

6.46.2. Содержимое пакета Kmod

Установленные программы: depmod (link to kmod), insmod (link to kmod), kmod, lsmod (link to kmod), modinfo (link to kmod), modprobe (link to kmod), and rmmod (link to kmod)
Установленная библиотека: libkmod.so

Краткое описание

depmod Создает файл зависимостей на основе символов, которые он находит в существующем наборе модулей; этот файл зависимостей используется программой **modprobe** для автоматической загрузки необходимых модулей

insmod программа для вставки модуля в ядро Linux

kmod Загружает и выгружает модули ядра

lsmod Выводит список загруженных модулей

modinfo Изучает объектный файл, связанный с модулем ядра и отображает любую информацию, которую он может получить

modprobe Использует файл зависимостей, созданный **depmod**, для автоматической загрузки соответствующих модулей

rmmod Выгружает модули из работающего ядра

`libkmod` Эта библиотека используется другими программами для загрузки и выгрузки модулей ядра

6.47. Gettext-0.20.1

Пакет содержит утилиты и библиотеки для работы с локализацией и интернационализацией необходимые для некоторых пакетов. Что позволяет программам, которые скомпилированы с поддержкой NLS (Поддержка нативных языков), показывать сообщения в пользовательском нативном языке.

Приблизительное время сборки: 2.9 SBU
Требуемое дисковое пространство: 249 MB

6.47.1. Установка пакета Gettext

Подготовьте пакет Gettext к компиляции:

```
./configure --prefix=/usr    \
            --disable-static \
            --docdir=/usr/share/doc/gettext-0.20.1
```

Скомпилируйте пакет:

```
make
```

Для выполнения тестов (выполнение может занять много времени, примерно 3 SBU), выполните следующую команду:

```
make check
```

Установите пакет:

```
make install
chmod -v 0755 /usr/lib/preloadable_libintl.so
```

6.47.2. 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, and xgettext
Установленные библиотеки: libasprintf.so, libgettextlib.so, libgettextpo.so, libgettextsrc.so, and preloadable_libintl.so
Установленные каталоги: /usr/lib/gettext, /usr/share/doc/gettext-0.20.1, /usr/share/gettext, and /usr/share/gettext-0.20.1

Краткое описание

autopoint Копирует стандартные файлы инфраструктуры Gettext в исходный код пакета

envsubst Заменяет переменные среды в строках формата оболочки

gettext	Переводит сообщение на естественном языке на язык пользователя поиск перевода в каталоге сообщений
gettext.sh	В первую очередь служит в качестве библиотеки функций оболочки для gettext
gettextize	Копирует все стандартные файлы Gettext в заданный верхний уровень каталог пакета для его интернационализации
msgattrib	Фильтрует сообщения каталога переводов по их атрибутам и манипулирует атрибутами
msgcat	Объединяет переданные файлы `.po`
msgcmp	Сравнивает два файла `.po` для проверки, что оба содержат один и тот же набор строк msgid
msgcomm	Находит сообщения, которые являются общими для файлов `.po` files
msgconv	Преобразует каталог переводов в иную кодировку символов
msgen	Создает английский каталог переводов
msgexec	Применяет команду ко всем переводам каталога
msgfilter	Применяет фильтр ко всем переводам каталога
msgfmt	Создает двоичный каталог сообщений из транслируемого каталога
msggrep	Извлекает все сообщения каталога переводов, которые соответствуют шаблону или принадлежат к некоторым заданным исходным файлам
msginit	Создаёт новый `.po` файл, инициализирующий метаинформацию со значениями из пользовательской среды
msgmerge	Объединяет два необработанных перевода в один файл
msgunfmt	Декомпилирует каталог бинарных сообщений в исходный текст перевода
msguniq	Унифицирует дубликаты переводов в каталоге переводов
ngettext	Отображает переводы на родном языке текстового сообщения, грамматическая форма зависит от числа
recode-sr-latin	Редактирует сербский текст с кириллицы на латиницу
xgettext	Извлекает переведенные строки сообщений из данного источника файлов для создания первого шаблона перевода
`libasprintf`	Определяет класс *autosprintf*, который создает Си-форматированный вывод полезный для работы в программах на языке Си++, для использования со <*string*> строками и <*iostream*> потоками
`libgettextlib`	частная библиотека, содержаща общие процедуры, используемые различными программами Gettext; они не предназначены для общего использования
`libgettextpo`	Используется для написания специализированных программ, которые обрабатывают файлы `.po` ; эта библиотека используется, когда стандартных приложений, поставляемых с Gettext (таких как **msgcomm**, **msgcmp**, **msgattrib**, и **msgen**) будет недостаточно

libgettextsrc	Частная библиотека, содержащая общие подпрограммы, используемые различными программами Gettext; они не предназначены для общего использования
preloadable_libintl	Библиотека, предназначенная для использования LD_PRELOAD, которая помогает libintl в журналировании непереведённых сообщений

6.48. Libelf из пакета Elfutils-0.177

Libelf - библиотека для обработки файлов формата ELF (Executable and Linkable Format — формат исполнимых и компонуемых файлов). формат двоичных файлов, используемый во многих современных UNIX-подобных операционных системах, таких как FreeBSD, Linux, Solaris и др. Также этот формат используется и во многих других системах. Большинство утилит доступны в других пакетах, но эта библиотека необходима для сборки ядра Linux используя конфигурацию по умолчанию (и наиболее эффективную).

Приблизительное время сборки: 1.1 SBU
Требуемое дисковое пространство: 95 MB

6.48.1. Установка пакета Libelf

Libelf является частью пакета elfutils-0.177. Необходимо использовать файл elfutils-0.177.tar.bz2 как архив с исходными текстами.

Подготовьте Libelf к компиляции:

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

Скомпилируйте пакет:

```
make
```

Для выполнения набора тестов, выполните команду:

```
make check
```

Выполните установку только Libelf:

```
make -C libelf install
install -vm644 config/libelf.pc /usr/lib/pkgconfig
```

6.48.2. Содержимое пакета Libelf

Установленная библиотека: libelf.so
Установленный каталог: /usr/include/elfutils

6.49. Libffi-3.2.1

Переносимый, высокоуровневый интерфейс по различным соглашениям о вызовах. Программы во время компиляции могу не знать об аргументах, которые были переданы функции. Например, интерпретатору можно указать во время выполнения количество аргументов и указать их тип, для вызова функции. Libffi может использоваться в программах как "мост" от интерпретатора к скомпилированному коду.

Приблизительное время сборки: 0.4 SBU
Требуемое дисковое пространство: 7.6 MB

6.49.1. Установка пакета Libffi

Note

Подобно GMP, libffi собирается с оптимизацией, специфичной для процессора. Если происходит сборка для другой системы, установите значения для CFLAGS и CXXFLAGS, чтобы указать общую сборку для вашей архитектуры. Если этого не сделать, все приложения, которые ссылаются на libffi, выдадут ошибку Illegal Operation Errors.

Модифицируйте Makefile чтобы установить заголовочные файлы в стандартный каталог `/usr/include` вместо `/usr/lib/libffi-3.2.1/include` .

```
sed -e '/^includesdir/ s/$(libdir).*$/$(includedir)/' \
    -i include/Makefile.in

sed -e '/^includedir/ s/=.*$/=@includedir@/' \
    -e 's/^Cflags: -I${includedir}/Cflags:/' \
    -i libffi.pc.in
```

Подготовьте пакет libffi к компиляции:

```
./configure --prefix=/usr --disable-static --with-gcc-arch=native
```

Значения параметров конфигурации:

`--with-gcc-arch=native`
Убедитесь, что GCC оптимизирован для текущей системы. Если аргумент явно не указан, то система будет определена автоматически, и генерируемый код может быть неправильным для некоторых систем. Если сгенерированный код будет скопирован из системы в менее совместимую, задайте её в качестве параметра. Подробнее об альтернативных типах систем см. *параметры x86 в руководстве GCC* .

Скомпилируйте пакет:

```
make
```

Для выполнения тестов, выполните команду:

```
make check
```

Установите пакет:

```
make install
```

6.49.2. Содержимое пакета Libffi

Установленная библиотека: libffi.so

Краткое описание

`libffi` Содержит libffi API функции

6.50. OpenSSL-1.1.1c

The OpenSSL пакет содержит инструменты управления и библиотеки криптографии. Они полезны для предоставления криптографических функций к другим пакетам, таким как OpenSSH, почтовым приложениям и веб-браузерам (для доступа к сайтам HTTPS).

Приблизительное время сборки: 2.3 SBU
Требуемое дисковое пространство: 147 MB

6.50.1. Установка пакета OpenSSL

Во-первых, устраните проблему, обнаруженную в новых версиях:

```
sed -i '/\} data/s/ =.*$/;\n    memset(\&data, 0, sizeof(data));/' \
  crypto/rand/rand_lib.c
```

Подготовьте пакет OpenSSL к компиляции:

```
./config --prefix=/usr         \
         --openssldir=/etc/ssl \
         --libdir=lib          \
         shared                \
         zlib-dynamic
```

Скомпилируйте пакет:

```
make
```

Для выполнения тестов, выполните команду:

```
make test
```

Установите пакет:

```
sed -i '/INSTALL_LIBS/s/libcrypto.a libssl.a//' Makefile
make MANSUFFIX=ssl install
```

При желании, можно установить документацию:

```
mv -v /usr/share/doc/openssl /usr/share/doc/openssl-1.1.1c
cp -vfr doc/* /usr/share/doc/openssl-1.1.1c
```

6.50.2. Содержимое пакета OpenSSL

Установленные программы: c_rehash and openssl
Установленные библиотеки: libcrypto.{so,a} and libssl.{so,a}
Установленные каталоги: /etc/ssl, /usr/include/openssl, /usr/lib/engines and /usr/share/doc/openssl-1.1.1c

Краткое описание

c_rehash Сценарий Perl, который сканирует все файлы в каталоге и добавляет символические ссылки к их хэш-значениям

openssl Инструмент командной строки для использования различных криптографических функций библиотеки OpenSSL из оболочки. Его можно использовать для различных функций которые документированы в **man 1 openssl**.

`libcrypto.so` реализует широкий спектр криптографических алгоритмов, используемых в различных Internet стандартах. Сервисы, предоставляемые библиотекой используются OpenSSL реализацией SSL, TLS и S/MIME, и также используются в реализации OpenSSH, OpenPGP, и других криптографических стандартах

`libssl.so` Реализация протокола защиты транспортного уровня (TLS v1). Предоставляет функции API, документацию с которой можно ознакомиться, выполнив команду **man 3 ssl**.

6.51. Python-3.7.4

Python 3 - Высокоуровневый язык программирования общего назначения, ориентированный на повышение производительности разработчика и читаемости кода. Содержит среду разработки для объектно-ориентированного программирования, написания сценариев, прототипирования крупных программ и разработки полноценных приложений.

Приблизительное время сборки: 1.3 SBU
Требуемое дисковое пространство: 399 MB

6.51.1. Установка пакета Python 3

Подготовьте пакет Python к компиляции:

```
./configure --prefix=/usr       \
            --enable-shared     \
            --with-system-expat \
            --with-system-ffi   \
            --with-ensurepip=yes
```

Значение параметров конфигурации:

`--with-system-expat`
Значение аргумента позволяет установить связь с системной библиотекой Expat.

`--with-system-ffi`
Значение аргумента позволяет установить связь с системной библиотекой libffi.

`--with-ensurepip=yes`
Значение аргумента включает сборку **pip** и **setuptools** программ для работы с пакетами.

Скомпилируйте пакет:

```
make
```

Для выполнения тестов необходимы TK и X Windows сессия, и они не могут быть выполнены до тех пор, пока пакет не будет переустановлен по инструкциям в книге BLFS.

Установите пакет:

```
make install
chmod -v 755 /usr/lib/libpython3.7m.so
chmod -v 755 /usr/lib/libpython3.so
ln -sfv pip3.7 /usr/bin/pip3
```

Значение команд установки:

chmod -v 755 /usr/lib/libpython3.{7m.,}so
Исправляет установку разрешений для библиотек для совместимости с другими библиотеками.

При желании установите заранее отформатированную документацию:

```
install -v -dm755 /usr/share/doc/python-3.7.4/html

tar --strip-components=1  \
    --no-same-owner       \
    --no-same-permissions \
    -C /usr/share/doc/python-3.7.4/html \
    -xvf ../python-3.7.4-docs-html.tar.bz2
```

Значения команд установки документации:

`--no-same-owner` and --no-same-permissions
Гарантирует что установленные файлы будут иметь правильные права владения. Без этих опций, при использовании программы tar при распаковке, будут установлены права на распакованные файлы с указанием владельца пакета.

6.51.2. Содержимое пакета Python 3

Установленные программы:	2to3, idle3, pip3, pydoc3, python3, python3-config, and pyvenv
Установленная библиотека:	libpython3.7m.so and libpython3.so
Установленные каталоги:	/usr/include/python3.7m, /usr/lib/python3 and /usr/share/doc/python-3.7.4

Краткое описание

2to3	Программа Python которая читает исходный код Python 2.x и применяет ряд исправлений для преобразования в правильный код Python 3.x.
idle3	Обёртка для сценария которая открывает графический редактор Python. Для запуска этого сценария убедитесь в том, что установлена программа Tk до запуска Python и модуль Tkinter также был собран.
pip3	Установщик пакетов для Python. Вы можете использовать pip для установки пакетов из индекса пакетов Python или других индексов.
pydoc3	Инструмент Python для работы с документацией.
python3	интерпретируемый, интерактивный, объектно-ориентированный язык программирования.
pyvenv	создает виртуальные Python среды в целевых каталогах.

6.52. Ninja-1.9.0

Ninja - утилита для сборки программ, фокусирующая на скорости. От других систем сборки программ она отличается двумя основными аспектами: для работы используется свой формат входных файлов, созданных системой построения более высокого уровня, и она предназначена для быстрой сборки программ.

Приблизительное время сборки: 0.2 SBU
Требуемое дисковое пространство: 69 MB

6.52.1. Установка пакета Ninja

Когда программа ninja запущена, как правило, используется максимальное количество процессов параллельно. По умолчанию этому числу присваивается значение количество ядер компьютера плюс два. В некоторых случаях это может привести к перегреву центрального процессора, и (или) переполнению оперативной памяти. Если выполнять запуск через командную строку, можно передать аргумент -jN, в котором можно явно указать количество потоков для выполнения. Но в некоторых пакетах данное значение встроено и аргумент -jN будет игнорирован.

Можно использовать *опциональный* патч, который позволит устанавливать количество потоков выполнения через переменную среды, NINJAJOBS. **Например** указание значения переменной среды:

```
export NINJAJOBS=4
```

ограничит одновременное выполнение ninja в четыре процесса.

If desired, add the capability to use the environment variable NINJAJOBS by running:

```
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
```

Выполните сборку пакета Ninja выполнив команду:

```
python3 configure.py --bootstrap
```

Значения аргументов сборки:

`--bootstrap`
Аргумент указывает ninja пересобрать себя в текущей системе.

Для выполнения тестов, выполните команду:

```
./ninja ninja_test
./ninja ninja_test
./ninja_test --gtest_filter=-SubprocessTest.SetWithLots
```

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
```

6.52.2. Содержимое пакета Ninja

Установленные программы: ninja

Краткое описание

ninja системы сборки ninja.

6.53. Meson-0.51.1

Meson - высокопроизводительная и удобная система сборки с открытым исходным кодом. Пакет содержит инструменты для автоматизации сборки программ. Основная цель Meson - минимизировать затраты времени на конфигурирование системы сборки.

Приблизительное время сборки: less than 0.1 SBU
Требуемое дисковое пространство: 28 MB

6.53.1. Установка пакета Meson

Для компиляции пакета Meson, необходимо выполнить следующую команду:

```
python3 setup.py build
```

У этого пакета нет тестов.

Установите пакет:

```
python3 setup.py install --root=dest
cp -rv dest/* /
```

Значения параметров установки:

`--root=dest`
По умолчанию команда **python3 setup.py install** выполняет установку различных файлов (например файлы руководств) в Python Eggs. Если указать значение dest в аргументе root, **setup.py** выполнит установку файлов в обычную иерархию каталогов. Затем мы можем скопировать иерархию, чтобы все файлы в итоге были в обычном месте.

6.53.2. Содержимое пакета Meson

Установленные программы: meson
Каталог установки: /usr/lib/python3.7/site-packages/meson-0.51.1-py3.7.egg-info and /usr/lib/python3.7/site-packages/mesonbuild

Краткое описание

meson Высокопроизводительная система сборки

6.54. Coreutils-8.31

Coreutils - пакет программного обеспечения GNU, содержащий большое количество основных утилит, таких как cat, ls и rm, необходимых для UNIX-подобных операционных систем. Пакет включает несколько более ранних пакетов — textutils, shellutils, fileutils и другие различные утилиты.

Приблизительное время сборки: 2.5 SBU
Требуемое дисковое пространство: 202 MB

6.54.1. Установка пакета Coreutils

POSIX требует, чтобы программы из Coreutils правильно распознавали границы символов даже в многобайтовых локалях. Следующий патч исправляет это несоблюдение и другие ошибки, связанные с интернационализацией.

```
patch -Np1 -i ../coreutils-8.31-i18n-1.patch
```

Note

Ранее в этом патче было обнаружено множество ошибок. Когда вы сообщаете о новых ошибках разработчикам, сопровождающим пакет Coreutils, сперва проверьте, воспроизводятся ли они без этого патча.

Уберем тест, который на некоторых машинах может привести к вечному зацикливанию:

```
sed -i '/test.lock/s/^/#/' gnulib-tests/gnulib.mk
```

Теперь, выполним подготовку пакета Coreutils к компиляции:

```
FORCE_UNSAFE_CONFIGURE=1 ./configure \
            --prefix=/usr            \
            --enable-no-install-program=kill,uptime
```

Значение параметров конфигурации:

`FORCE_UNSAFE_CONFIGURE=1`
Эта переменная среды позволяет выполнить сборку пакета от корневого пользователя root.

`--enable-no-install-program=kill,uptime`
Целью этого аргумента является предотвращение использования Coreutils установки двоичных файлов, которые позже будут установлены другими пакетами.

Скомпилируйте пакет:

```
make
```

Перейдите к разделу “установка пакета” если не планируете запускать тесты.

Сейчас, набор тестов готов к запуску. Сначала запустите те тесты, которые предназначены для запуска от корневого пользователя `root`:

```
make NON_ROOT_USERNAME=nobody check-root
```

Мы будем запускать оставшиеся тесты от пользователя nobody. Однако некоторые тесты требуют, чтобы пользователь был членом более чем одной группы. Чтобы эти тесты не были пропущены, мы добавим временную группу сделаем пользователя nobody её членом:

```
echo "dummy:x:1000:nobody" >> /etc/group
```

Исправьте некоторые разрешения, чтобы пользователь, не являющийся пользователем root, мог компилировать и выполять тесты:

```
chown -Rv nobody .
```

Теперь, запустим тесты. Убедитесь, что переменная окружения PATH в среде **su** содержит каталог /tools/bin.

```
su nobody -s /bin/bash \
          -c "PATH=$PATH make RUN_EXPENSIVE_TESTS=yes check"
```

Известно, что тест test-getlogin не пройдет в частично собранной системе, в среде chroot, но успешно выполнится в конце этой главы. Тестовая программа tty.sh также известно, что он не работает.

Удалим временную группу:

```
sed -i '/dummy/d' /etc/group
```

Установите пакет:

```
make install
```

Переместим программы в места, указанные в FHS:

```
mv -v /usr/bin/{cat,chgrp,chmod,chown,cp,date,dd,df,echo} /bin
mv -v /usr/bin/{false,ln,ls,mkdir,mknod,mv,pwd,rm} /bin
mv -v /usr/bin/{rmdir,stty,sync,true,uname} /bin
mv -v /usr/bin/chroot /usr/sbin
mv -v /usr/share/man/man1/chroot.1 /usr/share/man/man8/chroot.8
sed -i s/\"1\"/\"8\"/1 /usr/share/man/man8/chroot.8
```

Некоторые сценарии в пакете LFS-Bootscripts зависят от **head**, **sleep**, **nice** и **touch**. Так как каталог `/usr` может быть недоступен на ранних стадиях процесса загрузки, то эти бинарные файлы должны находится в корневом каталоге раздела для соблюдния совместимости с FHS:

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

6.54.2. 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, and yes

Установленная библиотека: libstdbuf.so (in /usr/libexec/coreutils)

Каталог установки: /usr/libexec/coreutils

Краткое описание

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 Отображает и проверяет BLAKE2 (512-bit) контрольную сумму

basename Удаляет любой путь и заданный суффикс из имени файла

basenc Кодирует и декодирует данные используя различные алгоритмы

cat Объединение файлов в стандартный вывод

chcon Изменяет контекст безопасности для файлов и каталогов

chgrp Изменяет групповое владение файлами и каталогами

chmod Изменяет разрешения каждого файла на заданный режим; режим может быть либо символическим представлением изменений или восьмеричное число, представляющее новые разрешения

chown Изменяет права пользователя и / или группы на файлы и каталоги

chroot Выполняет команду с указанным каталогом как корневой каталог /

cksum Показывает контрольную сумму (CRC) и количество байт для каждого указанного файла

comm Сравнивает два файла, выводящих в трех столбцах строки которые являются уникальными, и линии, которые являются общими

cp Выполняет копирование файлов

csplit Разделяет данный файл на несколько новых файлов, разделяя их в соответствии с заданными шаблонами или номерами строк и выводит количество байтов каждого нового файла

cut Распечатывает разделы линий, выбирая детали в соответствии с данными поля или позиции

date Отображает текущее время в данном формате или устанавливает системную дату

dd Копирует файл с использованием заданного размера и количества блоков, в то время как возможно выполнение конверсий с ним

df Показывает список всех файловых систем по именам устройств, сообщает их размер, занятое и свободное пространство и точки монтирования.

dir Выводит список содержимого каждого переданного каталога (тоже самое что команда **ls**)

dircolors Устанавливает цветовую схему и устанавливает значение переменной среды LS_COLOR для вывода команды **ls**

dirname Удаляет имя файла из полного пути

du Сообщает объем дискового пространства, используемого текущим каталогом, для каждого из указанных каталогов (включая все подкаталоги) или по каждому из указанных файлов

echo Отображает строку текста

env Выполняет команду в измененной среде

expand Преобразует знаки табуляции в пробелы

expr Вычисляет выражения

factor Раскладывает число на простые множители

false Ничего не выполняет, выходит со статусом, отображающим завершение со сбоем

fmt Переформатирует абзацы в указанных файлах

fold Разбивает длинные строки для устройств вывода с ограниченной шириной

groups Выводит группы, в которых состоит пользователь

head Выводит несколько первых строк файла (или заданное количество)

hostid Выводит цифровой идентификатор для текущего хоста (в шестнадцатеричном виде)

id Выводит реальный/эффективный UID и GID и группы, в которых состоит указанный пользователь

install Копирует файлы и устанавливает атрибуты

join Объединяет файлы по общему полю

link Вызывает функцию link() для создания жесткой ссылки на файл

ln Создаёт символическую ссылку на файл

logname Сообщает имя пользователя текущего пользователя

ls Перечисляет содержимое каждого заданного каталога

md5sum Вычисляет и проверяет хеш MD5

mkdir Создает каталог с указанным наименованием

mkfifo Создаёт FIFO (именованные каналы)

mknod Создает узлы устройства с указанными именами; узел устройства является символьным специальным файлом, специальным файлом блока или файлом FIFO

mktemp Создает временные файлы безопасным образом; он используется при работе различных сценариев

mv Перемещает и переименовывает файлы или каталоги

nice Запускает программу с указанным приоритетом

nl Количество строк у заданного файла

nohup Позволить команде выполняться после выхода пользователя (logout)

nproc Распечатывает количество блоков обработки, доступных для процесса

numfmt Преобразует числа в строки или наборот в удобочитаемом виде

od Выводит содержимое файлов в восьмеричном и других форматах

paste Слияние заданных файлов, объединение последовательно строк вместе, разделенных символами табуляции

pathchk Проверяет допустимость или переносимость имен файлов

pinky Легкий клиент сетевого протокола, предназначенного для предоставления информации о пользователях удалённого компьютера.; он сообщает некоторую информацию о данных пользователя

pr Преобразует текстовые файлы в формат для печати

printenv Выводит переменные окружения

printf Выводит заданные аргументы в соответствии с заданным форматом. Что то вроде функции языка Си printf

ptx Создает перестановочный индекс из содержимого данных файлов, с каждым ключевым словом в его контексте

pwd Выводит текущий рабочий каталог

readlink Выводит значение символической ссылки

realpath Возвращает полученный абсолютный или относительный путь к файлу

rm Удаляет файлы или каталоги

rmdir Удаляет пустые каталоги

runcon Запускает команду с указанным контекстом безопасности

seq Выводит числа по порядку с указанными диапазоном и шагом

sha1sum Вывод или проверка 160-битного алгоритма безопасного хеша 1 (SHA1) контрольной суммы

sha224sum Вывод или проверка 224-битного алгоритма безопасного хеша контрольной суммы

sha256sum Вывод или проверка 256-битного алгоритма безопасного хеша контрольной суммы

sha384sum Вывод или проверка 384-битного алгоритма безопасного хеша контрольной суммы

sha512sum Вывод или проверка 512-битного алгоритма безопасного хеша контрольной суммы

shred Перезаписывает файлы чтобы скрыть содержимое (так называемое безопасное удаление), и опционально удаляет файлы

shuf Перемешивает строки текста

sleep Пауза на заданное количество времени

sort Сортирует строки из заданных файлов

split Разделяет файл на куски, по размеру или по числу линий

stat Отображает статус файла или файловой системы

stdbuf Выполняет команды с измененными операциями буферизации для своего стандартного потока

stty Изменяет и выводит настройки терминала

sum Проверяет контрольные суммы файла

sync Сбрасывает буферы файловой системы; заставляет измененные блоки сохранить на диск и обновить суперблок

tac Объединение данных файлов в обратном порядке

tail Распечатывает последние десять строк (или заданное количество строк) каждого заданного файла

tee Отправляет вывод на множество файлов

test Сравнивает значения и проверяет типы файлов

timeout Выполняет команду с ограничением по времени

touch Изменяет временные метки файлов, устанавливая доступ и модификацию время данных файлов до текущего времени; файлов, которые не существуют создаются с нулевой длиной

tr Переводит, сжимает и удаляет данные из стандартного ввода

true Ничего не делает; всегда выходит со статусом успех

truncate Сжимает или расширяет файл до указанного размера

tsort Выполняет топологическую сортировку; он пишет полностью упорядоченный список в соответствии с частичным упорядочением в заданном файле

tty Сообщает имя файла терминала, подключенного к стандартному вводу

uname Информация о системе

unexpand Преобразует пробелы в знаки табуляции

uniq Удаляет повторяющиеся строки из упорядоченного файла

unlink Вызывает функцию unlink() для удаления заданных файлов

users Сообщает имена пользователей, которые в настоящее время вошли в систему

vdir Тоже самое что и команда **ls -l**

wc Сообщает количество строк, слов и байтов для каждого заданного файл, а также общую строку, когда дано более одного файла

who Выводит список всех вошедших пользователей

whoami Сообщает имя пользователя, связанное с действующим Идентификатором пользователя

yes До бесконечности выводит “y” или заданную строку, пока процесс не будет принудительно завершен

`libstdbuf` библиотека, используемая **stdbuf**

6.55. Check-0.12.0

Check - платформа для работы с модульными тестами для языка программирования Си

Приблизительное время сборки: 0.1 SBU (about 3.7 SBU with tests)
Требуемое дисковое пространство: 12 MB

6.55.1. Установка пакета Check

Подготовьте пакет Check к компиляции:

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

Выполните сборку пакета:

```
make
```

Когда компиляция будет завершена, можно запустить тесты, выполнив следующую команду:

```
make check
```

Обратите внимание, что тесты могут выполняться довольно продолжительное время (до 4 SBU).

Установите пакет и скорректируйте сценарий:

```
make docdir=/usr/share/doc/check-0.12.0 install
sed -i '1 s/tools/usr/' /usr/bin/checkmk
```

6.55.2. Содержимое пакета Check

Установленная программа: checkmk
Установленная библиотека: libcheck.{a,so}

Краткое описание

checkmk Сценарий Awk для создания модульных тестов Си для использования с Check

`libcheck.{a,so}` Содержит функции, позволяющие вызывать Check из программ для тестирования

6.56. Diffutils-3.7

Пакет содержит программы, которые отображают разницу в содержимом между файлами и каталогами. Эти программы могут быть использованы, для создания патчей, а также они используются в процедурах сборки для большинства пакетов.

Приблизительное время сборки: 0.4 SBU
Требуемое дисковое пространство: 36 MB

6.56.1. Установка пакета Diffutils

Подготовьте пакет Diffutils к компиляции:

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

Скомпилируйте пакет:

```
make
```

Для выполнения тестов, выполните команду:

```
make check
```

Установите пакет:

```
make install
```

6.56.2. Содержимое пакета Diffutils

Установленные программы: cmp, diff, diff3, and sdiff

Краткое описание

cmp Сравнивает два файла и сообщает, в каких байтах есть различия
diff Сравнивает два файла или каталога и сообщает, в каких строках есть различия
diff3 Сравнивает три файла, строка за строкой
sdiff Объединяет два файла и интерактивно выводит результаты

6.57. Gawk-5.0.1

Пакет содержит программы для манипуляции с текстовыми файлами. Это GNU версия awk (Aho-Weinberg-Kernighan). Он используется в процедурах сборки для большинства пакетов.

Приблизительное время сборки: 0.4 SBU
Требуемое дисковое пространство: 47 MB

6.57.1. Установка пакета Gawk

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

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

Подготовьте пакет Gawk к компиляции:

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

Скомпилируйте пакет:

```
make
```

Для выполнения тестов, выполните команду:

```
make check
```

Установите пакет:

```
make install
```

При желании, можно установить документацию:

```
mkdir -v /usr/share/doc/gawk-5.0.1
cp    -v doc/{awkforai.txt,*.{eps,pdf,jpg}} /usr/share/doc/gawk-5.0.1
```

6.57.2. Содержимое пакета Gawk

Установленные программы: awk (link to gawk), gawk, and awk-5.0.1
Установленные библиотеки: filefuncs.so, fnmatch.so, fork.so, inplace.so, intdiv.so, ordchr.so, readdir.so, readfile.so, revoutput.so, revtwoway.so, rwarray.so, and time.so (all in /usr/lib/gawk)
Установленные каталоги: /usr/lib/gawk, /usr/libexec/awk, /usr/share/awk, and /usr/share/doc/gawk-5.0.1

Краткое описание

awk Символическая ссылка на **gawk**

gawk программа для манипуляции с текстовыми файлами; GNU реализация программы **awk**

gawk-5.0.1 Жесткая ссылка на **gawk**

6.58. Findutils-4.6.0

The Findutils пакет содержит программы для поиска файлов. Программы предоставляют способы для рекурсивного поиска файлов по дереву каталогов, создания, обслуживания и поиска в базе данных (как правильно поиск выполняется быстрее, чем рекурсивный поиск, но менее надёжен, если база данных не в актуальном состоянии).

Приблизительное время сборки: 0.7 SBU
Требуемое дисковое пространство: 52 MB

6.58.1. Установка пакета Findutils

Уберем тест, который на некоторых машинах может привести к вечному зацикливанию:

```
sed -i 's/test-lock..EXEEXT.//' tests/Makefile.in
```

Далее, сделаем некоторые исправления, требуемые glibc-2.28:

```
sed -i 's/IO_ftrylockfile/IO_EOF_SEEN/' gl/lib/*.c
sed -i '/unistd/a #include <sys/sysmacros.h>' gl/lib/mountlist.c
echo "#define _IO_IN_BACKUP 0x100" >> gl/lib/stdio-impl.h
```

Подготовьте пакет Findutils к компиляции:

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

Значение параметров конфигурации:

`--localstatedir`
Этот аргумент изменяет местоположение базы данных **locate** на каталог `/var/lib/locate` , для совместимости с FHS.

Скомпилируйте пакет:

```
make
```

Для выполнения тестов, выполните команду:

```
make check
```

Установите пакет:

```
make install
```

Некоторые сценарии в пакете LFS-Bootscripts зависят от **find**. Так как каталог `/usr` может быть не доступен на ранних стадиях загрузки системы, эти программы должны быть размещены в корневом каталоге раздела. Сценарий **updatedb** также должен быть модифицирован для указания правильного пути:

```
mv -v /usr/bin/find /bin
sed -i 's|find:=${BINDIR}|find:=/bin|' /usr/bin/updatedb
```

6.58.2. Содержимое пакета Findutils

Установленные программы: find, locate, updatedb, and xargs

Установленные каталоги: /var/lib/locate

Краткое описание

find Ищет файлы в указанном каталоге, по указанным критериям

locate быстрый поиск файлов в одной или нескольких базах данных имён файлов с использованием заданных пользователем критериев.

updatedb создание и/или обновление баз данных, используемых программой locate.

xargs формирование и исполнение команд на основании данных, считанных из стандартного ввода.

6.59. Groff-1.22.4

Пакет Groff содержит программы для обработки и форматирования текста. Одна из самых важных функций - форматирование man страниц.

Приблизительное время сборки: 0.5 SBU
Требуемое дисковое пространство: 95 MB

6.59.1. Установка пакета Groff

Groff ожидает переменную окружения PAGE значение которой должно содержать размер бумаги по умолчанию. Для пользователей из США уместным будет указать значение переменной *PAGE=letter*. В ином месте указание значение переменной *PAGE=A4* может быть более подходящим. Хотя значение по умолчанию размер бумаги задается во время компиляции, ее можно переопределить позже путем выполнения команды echo с указанием значения “A4” или “letter” в файле `/etc/papersize` .

Подготовьте пакет Groff к компиляции:

```
PAGE=<paper_size> ./configure --prefix=/usr
```

Этот пакет не поддерживает параллельную сборку. Выполните компиляцию пакета:

```
make -j1
```

У этого пакета нет тестов.

Установите пакет:

```
make install
```

6.59.2. 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, and troff
Установленные каталоги: /usr/lib/groff and /usr/share/doc/groff-1.22.4, /usr/share/groff

Краткое описание

addftinfo Выполняет чтение файла шрифта troff и добавляет некоторую дополнительную информацию по метрикам шрифта которая используется системой**groff**

afmtodit Создаёт файл шрифта для использования с **groff** и **grops**

chem Препроцессор Groff preprocessor для создания диаграмм химической структуры

eqn Составляет описания уравнений, встроенные во входные данные troff , которые понимаются **troff**

eqn2graph	Преобразует a troff EQN (уравнение) в обрезанное изображение
gdiffmk	Отображает различия между файлами groff/nroff/troff
glilypond	Преобразует ноты, записанные на языке lilypond в язык groff
gperl	Препроцессор groff, позволяя добавлять perl-код в файлы groff
gpinyin	Препроцессор для groff, позволяя добавлять Китайский и Европейские языки Pinyin в файлы groff.
grap2graph	Преобразует диаграммы в обрезанное растровое изображение
grn	Препроцессор **groff** для файлов gremlin
grodvi	A driver for **groff** that produces TeX dvi format
groff	A front-end to the groff document formatting system; normally, it runs the **troff** program and a post-processor appropriate for the selected device
groffer	Displays groff files and man pages on X and tty terminals
grog	Reads files and guesses which of the **groff** options `-e` , `-man` , `-me` , `-mm` , `-ms` , `-p` , `-s` , and `-t` are required for printing files, and reports the **groff** command including those options
grolbp	Is a **groff** driver for Canon CAPSL printers (LBP-4 and LBP-8 series laser printers)
grolj4	Is a driver for **groff** that produces output in PCL5 format suitable for an HP LaserJet 4 printer
gropdf	Translates the output of GNU **troff** to PDF
grops	Translates the output of GNU **troff** to PostScript
grotty	Translates the output of GNU **troff** into a form suitable for typewriter-like devices
hpftodit	Creates a font file for use with **groff -Tlj4** from an HP-tagged font metric file
indxbib	Creates an inverted index for the bibliographic databases with a specified file for use with **refer**, **lookbib**, and **lkbib**
lkbib	Searches bibliographic databases for references that contain specified keys and reports any references found
lookbib	Prints a prompt on the standard error (unless the standard input is not a terminal), reads a line containing a set of keywords from the standard input, searches the bibliographic databases in a specified file for references containing those keywords, prints any references found on the standard output, and repeats this process until the end of input
mmroff	A simple preprocessor for **groff**
neqn	Formats equations for American Standard Code for Information Interchange (ASCII) output
nroff	A script that emulates the **nroff** command using **groff**
pdfmom	Is a wrapper around groff that facilitates the production of PDF documents from files formatted with the mom macros.
pdfroff	Creates pdf documents using groff
pfbtops	Translates a PostScript font in `.pfb` format to ASCII

pic	Compiles descriptions of pictures embedded within troff or TeX input files into commands understood by TeX or **troff**
pic2graph	Converts a PIC diagram into a cropped image
post-grohtml	Translates the output of GNU **troff** to HTML
preconv	Converts encoding of input files to something GNU **troff** understands
pre-grohtml	Translates the output of GNU **troff** to HTML
refer	Copies the contents of a file to the standard output, except that lines between *.[* and *.]* are interpreted as citations, and lines between *.R1* and *.R2* are interpreted as commands for how citations are to be processed
roff2dvi	Transforms roff files into DVI format
roff2html	Transforms roff files into HTML format
roff2pdf	Transforms roff files into PDFs
roff2ps	Transforms roff files into ps files
roff2text	Transforms roff files into text files
roff2x	Transforms roff files into other formats
soelim	Reads files and replaces lines of the form *.so file* by the contents of the mentioned *file*
tbl	Compiles descriptions of tables embedded within troff input files into commands that are understood by **troff**
tfmtodit	Creates a font file for use with **groff -Tdvi**
troff	Is highly compatible with Unix **troff**; it should usually be invoked using the **groff** command, which will also run preprocessors and post-processors in the appropriate order and with the appropriate options

6.60. GRUB-2.04

GRUB является эталонной реализацией загрузчика, соответствующего спецификации Multiboot и может загрузить любую совместимую с ней операционную систему. Среди них: Linux, FreeBSD, Solaris и многие другие. Кроме того, GRUB умеет по цепочке передавать управление другому загрузчику, что позволяет ему загружать Windows (через загрузчик NTLDR или bootmgr), MS-DOS, OS/2 и другие системы.

Приблизительное время сборки: 0.8 SBU

Требуемое дисковое пространство: 161 MB

6.60.1. Установка пакета GRUB

Подготовьте пакет GRUB к компиляции:

```
./configure --prefix=/usr          \
            --sbindir=/sbin        \
            --sysconfdir=/etc      \
            --disable-efiemu       \
            --disable-werror
```

Значения новых параметров конфигурации:

`--disable-werror`
: Аргумент позволяет завершить сборку с предупреждениями с более поздними версиями Flex.

`--disable-efiemu`
: Аргумент минимизирует что будет собрано, отключив функции и программы тестирования, которые не требуются для системы LFS.

Скомпилируйте пакет:

```
make
```

У этого пакета нет тестов.

Установите пакет:

```
make install
mv -v /etc/bash_completion.d/grub /usr/share/bash-completion/completions
```

Информация о том как обеспечить загрузку системы, с использованием GRUB, будет описано в разделе Section 8.4, “Использование GRUB для настройки процесса загрузки”.

6.60.2. 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, and grub-syslinux2cfg

Установленные каталоги: /usr/lib/grub, /etc/grub.d, /usr/share/grub, и /boot/grub (при первом запуске grub-install)

Краткое описание

grub-bios-setup Is a helper program for grub-install

grub-editenv A tool to edit the environment block

grub-file Checks if FILE is of the specified type.

grub-fstest Tool to debug the filesystem driver

grub-glue-efi Processes ia32 and amd64 EFI images and glues them according to Apple format.

grub-install Установите GRUB on your drive

grub-kbdcomp Script that converts an xkb layout into one recognized by GRUB

grub-macbless Mac-style bless on HFS or HFS+ files

grub-menulst2cfg Converts a GRUB Legacy `menu.lst` into a `grub.cfg` for use with GRUB 2

grub-mkconfig Generate a grub config file

grub-mkimage Make a bootable image of GRUB

grub-mklayout Generates a GRUB keyboard layout file

grub-mknetdir Подготовьте пакеть a GRUB netboot directory

grub-mkpasswd-pbkdf2 Generates an encrypted PBKDF2 password for use in the boot menu

grub-mkrelpath Makes a system pathname relative to its root

grub-mkrescue Make a bootable image of GRUB suitable for a floppy disk or CDROM/DVD

grub-mkstandalone Generates a standalone image

grub-ofpathname Is a helper program that prints the path of a GRUB device

grub-probe Probe device information for a given path or device

grub-reboot Sets the default boot entry for GRUB for the next boot only

grub-render-label Render Apple .disk_label for Apple Macs

grub-script-check Checks GRUB configuration script for syntax errors

grub-set-default Sets the default boot entry for GRUB

grub-sparc64-setup Is a helper program for grub-setup

grub-syslinux2cfg Transform a syslinux config file into grub.cfg format

6.61. Less-551

Программа Less используется для просмотра (но не изменения) содержимого текстовых файлов на экране.Она также используется пакетом Man-DB для просмотра страниц руководств.

Приблизительное время сборки: less than 0.1 SBU
Требуемое дисковое пространство: 4.1 MB

6.61.1. Установка пакета Less

Подготовьте пакет Less к компиляции:

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

Значение параметров конфигурации:

`--sysconfdir=/etc`
This option tells the programs created by the package to look in `/etc` for the configuration files.

Скомпилируйте пакет:

```
make
```

У этого пакета нет тестов.

Установите пакет:

```
make install
```

6.61.2. Содержимое пакета Less

Установленные программы: less, lessecho, and lesskey

Краткое описание

less Просмотрщик файлов; он отображает содержимое указанного файла, позволяя пользователю выполнять навигацию по файлу, находить строки и переходить к заметкам

lessecho Необходим для расширения мета-символов, таких как * и *?* в наименованиях файлов систем Unix

lesskey Используется для указания привязок клавиш для**less**

6.62. Gzip-1.10

Пакет содержит программы для сжатия и распаковки файлов. Он необходим, чтобы выполнять распаковку большинства пакетов LFS.

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

6.62.1. Установка пакета Gzip

Подготовьте пакет Gzip к компиляции:

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

Скомпилируйте пакет:

```
make
```

Для выполнения тестов, выполните команду:

```
make check
```

Как известно, два теста не пройдут в среде LFS: help-version и zmore.

Установите пакет:

```
make install
```

Переместите программу которая должна находиться в корневой файловой системе:

```
mv -v /usr/bin/gzip /bin
```

6.62.2. Содержимое пакета Gzip

Установленные программы: gunzip, gzexe, gzip, uncompress (hard link with gunzip), zcat, zcmp, zdiff, zegrep, zfgrep, zforce, zgrep, zless, zmore, and znew

Краткое описание

gunzip Распаковывает сжатые файлы
gzexe Создает самораспаковывающиеся исполняемые файлы
gzip Сжимает данные файлы с помощью алгоритма Lempel-Ziv (LZ77)
uncompress Распаковывает сжатые файлы
zcat Распаковывает GZIP-файлы в стандартный вывод
zcmp Запускает программу **cmp** для сжатых файлов
zdiff Запускает программу **diff** для сжатых файлов
zegrep Запускает программу **egrep** для сжатых файлов
zfgrep Запускает программу **fgrep** для сжатых файлов

zforce присваивает расширение .gz всем архивам gzip, чтобы gzip не сжимал их дважды. Это может быть полезно для файлов, чьи имена были усечены при перемещении файлов. Для файловых систем с максимальной длиной имени файла в 14 символов исходное имя укорачивается, чтобы вместить суффикс .gz. Например, файл 12345678901234 переименовывается в 12345678901.gz. Файлы с именами, подобными файл.tgz, не затрагиваются.

zgrep Запускает программу **grep** для сжатых файлов

zless Запускает программу **less** для сжатых файлов

zmore Запускает программу **more** для сжатых файлов

znew перепаковывает файлы из формата .Z (compress) в формат .gz (gzip). Если вы хотите перепаковать файл, уже находящийся в формате gzip, присвойте ему расширение .Z и выполните znew.

6.63. IPRoute2-5.2.0

Iproute2 — это набор программ для управления параметрами сетевых устройств в ядре Linux. Эти утилиты были разработаны в качестве унифицированного интерфейса к ядру Linux, которое непосредственно управляет сетевым трафиком. Iproute2 заменил полный набор классических сетевых утилит UNIX, которые ранее использовались для настройки сетевых интерфейсов, таблиц маршрутизации и управления arp#таблицами: ifconfig, route, arp, netstat и других, предназначенных для создания IP#туннелей. iproute2 предлагает унифицированный синтаксис для управления самыми разными аспектами сетевых интерфейсов.

Приблизительное время сборки: 0.2 SBU
Требуемое дисковое пространство: 13 MB

6.63.1. Установка пакета IPRoute2

Программа **arpd** в этом пакете не будет собрана, так как зависит от Berkeley DB, которая не будет устанавливаться в LFS. Однако, каталог для **arpd** и страницы руководств будут установлены. Предотвратите это, выполнив приведенные ниже команды. Если бинарный файл **arpd** необходим, инструкции по установке пакета Berkeley DB можно найти в книге BLFS по ссылке *https://linuxfromscratch.ru/blfs/view/svn/server/databases.html#db*.

```
sed -i /ARPD/d Makefile
rm -fv man/man8/arpd.8
```

Также необходимо запретить сборку двух модулей, для которых необходим пакет *https://linuxfromscratch.ru/blfs/view/svn/postlfs/iptables.html*.

```
sed -i 's/.m_ipt.o//' tc/Makefile
```

Скомпилируйте пакет:

```
make
```

В этом пакете нет рабочих тестов

Установите пакет:

```
make DOCDIR=/usr/share/doc/iproute2-5.2.0 install
```

6.63.2. Содержимое пакета IPRoute2

Установленные программы: bridge, ctstat (link to lnstat), genl, ifcfg, ifstat, ip, lnstat, nstat, routef, routel, rtacct, rtmon, rtpr, rtstat (link to lnstat), ss, and tc
Установленные каталоги: /etc/iproute2, /usr/lib/tc, and /usr/share/doc/iproute2-5.2.0,

Краткое описание

bridge Configures network bridges

ctstat Connection status utility

genl Generic netlink utility frontend

ifcfg A shell script wrapper for the **ip** command [Note that it requires the **arping** and **rdisk** programs from the iputils package found at *http://www.skbuff.net/iputils/*.]

ifstat Shows the interface statistics, including the amount of transmitted and received packets by interface

ip The main executable. It has several different functions:
ip link *<device>* allows users to look at the state of devices and to make changes
ip addr allows users to look at addresses and their properties, add new addresses, and delete old ones
ip neighbor allows users to look at neighbor bindings and their properties, add new neighbor entries, and delete old ones
ip rule allows users to look at the routing policies and change them
ip route allows users to look at the routing table and change routing table rules
ip tunnel allows users to look at the IP tunnels and their properties, and change them
ip maddr allows users to look at the multicast addresses and their properties, and change them
ip mroute allows users to set, change, or delete the multicast routing
ip monitor allows users to continuously monitor the state of devices, addresses and routes

lnstat Provides Linux network statistics; it is a generalized and more feature-complete replacement for the old **rtstat** program

nstat Shows network statistics

routef A component of **ip route**. This is for flushing the routing tables

routel A component of **ip route**. This is for listing the routing tables

rtacct Displays the contents of `/proc/net/rt_acct`

rtmon Route monitoring utility

rtpr Converts the output of **ip -o** back into a readable form

rtstat Route status utility

ss Similar to the **netstat** command; shows active connections

tc Traffic Controlling Executable; this is for Quality Of Service (QOS) and Class Of Service (COS) implementations
tc qdisc allows users to setup the queueing discipline
tc class allows users to setup classes based on the queuing discipline scheduling
tc estimator allows users to estimate the network flow into a network
tc filter allows users to setup the QOS/COS packet filtering
tc policy allows users to setup the QOS/COS policies

6.64. Kbd-2.2.0

key-table файлы, утилиты для клавиатуры для не US наборов, а также консольные шрифты.

Приблизительное время сборки: 0.2 SBU
Требуемое дисковое пространство: 36 MB

6.64.1. Установка пакета Kbd

Поведение клавиш Backspace и Delete несовместимо в раскладке пакета Kbd. Следующий патч исправляет это для раскладок i386:

```
patch -Np1 -i ../kbd-2.2.0-backspace-1.patch
```

После исправления ключ Backspace генерирует символ с кодом 127, и клавиша Delete генерирует известную управляющую последовательность.

Удалить избыточную программу **resizecons** (требует отсутствующей библиотеки svgalib для предоставления файлов видеорежима - для нормального использования **setfont** в консоли) вместе с страницами руководства.

```
sed -i 's/\(RESIZECONS_PROGS=\)yes/\1no/g' configure
sed -i 's/resizecons.8 //' docs/man/man8/Makefile.in
```

Подготовьте пакет Kbd к компиляции:

```
PKG_CONFIG_PATH=/tools/lib/pkgconfig ./configure --prefix=/usr --disable-vlock
```

Значение параметров конфигурации:

`--disable-vlock`
Этот аргумент запрещает создание встроенной утилиты vlock. для которой требуется библиотека PAM, недоступной в среде chroot.

Скомпилируйте пакет:

```
make
```

Для выполнения тестов, выполните команду:

```
make check
```

Установите пакет:

```
make install
```

Note

For some languages (e.g., Belarusian) the Kbd package doesn't provide a useful keymap where the stock “by” keymap assumes the ISO-8859-5 encoding, and the CP1251 keymap is normally used. Users of such languages have to download working keymaps separately.

При желании, можно установить документацию:

```
mkdir -v       /usr/share/doc/kbd-2.2.0
cp -R -v docs/doc/* /usr/share/doc/kbd-2.2.0
```

6.64.2. Содержимое пакета Kbd

Установленные программы: chvt, deallocvt, dumpkeys, fgconsole, getkeycodes, kbdinfo, kbd_mode, kbdrate, loadkeys, loadunimap, mapscrn, openvt, psfaddtable (link to psfxtable), psfgettable (link to psfxtable), psfstriptable (link to psfxtable), psfxtable, setfont, setkeycodes, setleds, setmetamode, setvtrgb, showconsolefont, showkey, unicode_start, and unicode_stop

Установленные каталоги: /usr/share/consolefonts, /usr/share/consoletrans, /usr/share/keymaps, /usr/share/doc/kbd-2.2.0, and /usr/share/unimaps

Краткое описание

chvt Changes the foreground virtual terminal

deallocvt Deallocates unused virtual terminals

dumpkeys Dumps the keyboard translation tables

fgconsole Prints the number of the active virtual terminal

getkeycodes Prints the kernel scancode-to-keycode mapping table

kbdinfo Obtains information about the status of a console

kbd_mode Reports or sets the keyboard mode

kbdrate Sets the keyboard repeat and delay rates

loadkeys Loads the keyboard translation tables

loadunimap Loads the kernel unicode-to-font mapping table

mapscrn An obsolete program that used to load a user-defined output character mapping table into the console driver; this is now done by **setfont**

openvt Starts a program on a new virtual terminal (VT)

psfaddtable Adds a Unicode character table to a console font

psfgettable Extracts the embedded Unicode character table from a console font

psfstriptable Removes the embedded Unicode character table from a console font

psfxtable Handles Unicode character tables for console fonts

setfont Changes the Enhanced Graphic Adapter (EGA) and Video Graphics Array (VGA) fonts on the console

setkeycodes Loads kernel scancode-to-keycode mapping table entries; this is useful if there are unusual keys on the keyboard

setleds Sets the keyboard flags and Light Emitting Diodes (LEDs)

setmetamode Defines the keyboard meta-key handling

setvtrgb Sets the console color map in all virtual terminals

showconsolefont Shows the current EGA/VGA console screen font

showkey	Reports the scancodes, keycodes, and ASCII codes of the keys pressed on the keyboard
unicode_start	Puts the keyboard and console in UNICODE mode [Don't use this program unless your keymap file is in the ISO-8859-1 encoding. For other encodings, this utility produces incorrect results.]
unicode_stop	Reverts keyboard and console from UNICODE mode

6.65. Libpipeline-1.5.1

Библиотека для манипулирования работы с подпроцессами гибким и удобным способом.

Приблизительное время сборки: 0.1 SBU
Требуемое дисковое пространство: 9.0 MB

6.65.1. Установка пакета Libpipeline

Подготовьте пакет Libpipeline к компиляции:

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

Скомпилируйте пакет:

```
make
```

Для выполнения тестов, выполните команду:

```
make check
```

Установите пакет:

```
make install
```

6.65.2. Содержимое пакета Libpipeline

Установленная библиотека: libpipeline.so

Краткое описание

`libpipeline` Эта библиотека используется для безопасного построения подпроцессов

6.66. Make-4.2.1

Пакет содержит программы которые автоматизируют процесс преобразования файлов из одной формы в другую. Чаще всего это компиляция исходного кода в объектные файлы и последующая компоновка в исполняемые файлы или библиотеки. Он необходим для сборки пакетов LFS.

Приблизительное время сборки: 0.6 SBU
Требуемое дисковое пространство: 13 MB

6.66.1. Установка пакета Make

Необходимо обойти ошибку, вызванную glibc-2.27 и в более поздних версиях:

```
sed -i '211,217 d; 219,229 d; 232 d' glob/glob.c
```

Подготовьте пакет Make к компиляции:

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

Скомпилируйте пакет:

```
make
```

Набор тестов должен знать, где находятся файлы perl. Для этого будем использовать переменную окружения. Для выполнения тестов, выполните следующую команду:

```
make PERL5LIB=$PWD/tests/ check
```

Установите пакет:

```
make install
```

6.66.2. Содержимое пакета Make

Установленная программа: make

Краткое описание

make Автоматически определяет, какие части пакета должны быть (повторно) скомпилированы, и затем выдает соответствующие команды

6.67. Patch-2.7.6

Пакет содержит программу предназначенную для переноса правок (изменений) между разными версиями текстовых файлов. Информация о правке обычно содержится в отдельном файле, называемом "заплаткой" (*patch*), "правкой" или "файлом правки" (англ. patch file). Подобный файл, как правило, создается с помощью другой утилиты Unix — diff, позволяющей автоматически извлечь информацию о различиях в тексте файлов. Он необходим для выполнения сборки некоторых пакетов LFS.

Приблизительное время сборки: 0.2 SBU
Требуемое дисковое пространство: 13 MB

6.67.1. Установка пакета Patch

Подготовьте пакет Patch к компиляции:

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

Скомпилируйте пакет:

```
make
```

Для выполнения тестов, выполните команду:

```
make check
```

Установите пакет:

```
make install
```

6.67.2. Содержимое пакета Patch

Установленная программа: patch

Краткое описание

patch Изменяет файлы в соответствии с файлом патча (исправления). [Патч-файл обычно представляет собой список различий, созданный с помощью программы **diff**. Применяя эти различия к исходным файлам, патч создает исправленные версии.]

6.68. Man-DB-2.8.6.1

Пакет Man-DB содержит программы для поиска и просмотра справочных страниц.

Приблизительное время сборки: 0.4 SBU
Требуемое дисковое пространство: 38 MB

6.68.1. Установка пакета Man-DB

Подготовьте пакет Man-DB к компиляции:

```
./configure --prefix=/usr                         \
            --docdir=/usr/share/doc/man-db-2.8.6.1 \
            --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 конфигурации:

`--disable-setuid`
запрещает задавать программе **man** setuid пользователю man.

`--enable-cache-owner=bin`
задает права доступа общесистемному кешу пользователю, который является владельцем каталога bin.

`--with-...`
Эти три аргумента используются для настройки программ по умолчанию. **lynx** текстовый веб-обозреватель (см. инструкции по установке в книге BLFS), **vgrind** преобразовывает исходные кода программ в входные данные Groff и ,**grap** полезен для набора графов в документах Groff. Программы **vgrind** и **grap** обычно нужны для просмотра справочных страниц. Они не входят в состав книг LFS или BLFS, но вы можете установить их самостоятельно после сборки LFS если при желании.

`--with-systemd...`
These parameters prevent installing unneeded systemd directories and files.

Скомпилируйте пакет:

```
make
```

Для выполнения тестов, выполните команду:

```
make check
```

Установите пакет:

```
make install
```

Remove an unwanted directory used for service files which would cause some BLFS packages to also install files there:

```
rm -rfv /lib/systemd
```

6.68.2. Страницы руководств LFS на других языках

В нижеприведенной таблице показывает кодировку, которая допускается при использовании в Man-DB страницах, расположенных в каталоге `/usr/share/man/<ll>` . Также, Man-DB правильно определит, если страницы руководств были установлены вручную в этот каталог в кодировке utf-8.

Table 6.1. Expected character encoding of legacy 8-bit manual pages

Язык (код)	Кодировка	Язык (код)	Кодировка
Danish (da)	ISO-8859-1	Croatian (hr)	ISO-8859-2
German (de)	ISO-8859-1	Hungarian (hu)	ISO-8859-2
English (en)	ISO-8859-1	Japanese (ja)	EUC-JP
Spanish (es)	ISO-8859-1	Korean (ko)	EUC-KR
Estonian (et)	ISO-8859-1	Lithuanian (lt)	ISO-8859-13
Finnish (fi)	ISO-8859-1	Latvian (lv)	ISO-8859-13
French (fr)	ISO-8859-1	Macedonian (mk)	ISO-8859-5
Irish (ga)	ISO-8859-1	Polish (pl)	ISO-8859-2
Galician (gl)	ISO-8859-1	Romanian (ro)	ISO-8859-2
Indonesian (id)	ISO-8859-1	Russian (ru)	KOI8-R
Icelandic (is)	ISO-8859-1	Slovak (sk)	ISO-8859-2
Italian (it)	ISO-8859-1	Slovenian (sl)	ISO-8859-2
Norwegian Bokmal (nb)	ISO-8859-1	Serbian Latin (sr@latin)	ISO-8859-2
Dutch (nl)	ISO-8859-1	Serbian (sr)	ISO-8859-5
Norwegian Nynorsk (nn)	ISO-8859-1	Turkish (tr)	ISO-8859-9
Norwegian (no)	ISO-8859-1	Ukrainian (uk)	KOI8-U
Portuguese (pt)	ISO-8859-1	Vietnamese (vi)	TCVN5712-1
Swedish (sv)	ISO-8859-1	Simplified Chinese (zh_CN)	GBK
Belarusian (be)	CP1251	Simplified Chinese, Singapore (zh_SG)	GBK
Bulgarian (bg)	CP1251	Traditional Chinese, Hong Kong (zh_HK)	BIG5HKSCS
Czech (cs)	ISO-8859-2	Traditional Chinese (zh_TW)	BIG5
Greek (el)	ISO-8859-7		

Note

Справочные страницы на языках, отсутствующих в списке выше, не поддерживаются.

6.68.3. Содержимое пакета Man-DB

Установленные программы:	accessdb, apropos (link to whatis), catman, lexgrog, man, mandb, manpath, and whatis
Установленные библиотеки:	libman.so and libmandb.so (both in /usr/lib/man-db)
Установленные каталоги:	/usr/lib/man-db, /usr/libexec/man-db, and /usr/share/doc/man-db-2.8.6.1

Краткое описание

accessdb	Dumps the **whatis** database contents in human-readable form
apropos	Searches the **whatis** database and displays the short descriptions of system commands that contain a given string
catman	Creates or updates the pre-formatted manual pages
lexgrog	Displays one-line summary information about a given manual page
man	Formats and displays the requested manual page
mandb	Creates or updates the **whatis** database
manpath	Displays the contents of $MANPATH or (if $MANPATH is not set) a suitable search path based on the settings in man.conf and the user's environment
whatis	Searches the **whatis** database and displays the short descriptions of system commands that contain the given keyword as a separate word
`libman`	Содержит run-time support for **man**
`libmandb`	Содержит run-time support for **man**

6.69. Tar-1.32

Этот пакет обеспечивает возможности архивирования и извлечения почти всех пакетов, используемых в LFS.

Приблизительное время сборки: 2.2 SBU
Требуемое дисковое пространство: 45 MB

6.69.1. Установка пакета Tar

Подготовьте пакет Tar к компиляции:

```
FORCE_UNSAFE_CONFIGURE=1  \
./configure --prefix=/usr \
            --bindir=/bin
```

Значение параметров конфигурации:

FORCE_UNSAFE_CONFIGURE=1
Аргумент явно указывает чтобы тесты для `mknod` выполнялись от имени корневого пользователя root. Как правило, опасно выполнять такие тесты от корневого пользователя, но так как тест запускается в системе, которая ещё до конца не построена, такое переопределение аргумента можно считать допустимым.

Скомпилируйте пакет:

```
make
```

Для выполнения тестов (около 3 SBU), выполните следующую команду:

```
make check
```

Установите пакет:

```
make install
make -C doc install-html docdir=/usr/share/doc/tar-1.32
```

6.69.2. Содержимое пакета Tar

Установленные программы: tar
Каталог установки: /usr/share/doc/tar-1.32

Краткое описание

tar Создает, извлекает файлы и перечисляет содержимое архивов, также известный как тарболы

6.70. Texinfo-6.6

Этот пакет - система документирования и язык разметки, позволяющие создавать документы в разных форматах из одного исходного текста. Он используется в процедурах установки многих пакетов LFS.

Приблизительное время сборки: 0.8 SBU

Требуемое дисковое пространство: 122 MB

6.70.1. Установка пакета Texinfo

Подготовьте пакет Texinfo к компиляции:

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

Значение параметров конфигурации:

`--disable-static`
В этом случае сценарий настройки верхнего уровня будет выдавать предупреждение, что это нераспознанный параметр, но сценарий настройки для XSParagraph распознает его и использует его для отключения установки статического `XSParagraph.a` в каталоге `/usr/lib/texinfo` .

Скомпилируйте пакет:

```
make
```

Для выполнения тестов, выполните команду:

```
make check
```

Установите пакет:

```
make install
```

При необходимости установите компоненты, входящие в установку TeX:

```
make TEXMF=/usr/share/texmf install-tex
```

Значение параметров make:

`TEXMF=/usr/share/texmf`
В значении переменной `TEXMF` make-файла хранится корневой путь к дереву TeX, в случае дальнейшей установки этого пакета, позднее.

Система документации Info использует простой текстовый файл для хранения списка пунктов меню. Файл находится в каталоге `/usr/share/info/dir`. К сожалению, из-за случайных проблем в Make-файлах различных пакетов, он иногда может выйти из синхронизации с информационными страницами, установленными в системе. Если каталог `/usr/share/info/dir` когда-либо потребуется создать заново, следующие команды выполнят эту задачу:

```
pushd /usr/share/info
rm -v dir
for f in *
  do install-info $f dir 2>/dev/null
done
popd
```

6.70.2. Содержимое пакета Texinfo

Установленные программы: info, install-info, makeinfo (link to texi2any), pdftexi2dvi, pod2texi, texi2any, texi2dvi, texi2pdf, и texindex
Установленная библиотека: MiscXS.so, Parsetexi.so, and XSParagraph.so (Всё содержимое каталога /usr/lib/texinfo)
Установленные каталоги: /usr/share/texinfo and /usr/lib/texinfo

Краткое описание

info Used to read info pages which are similar to man pages, but often go much deeper than just explaining all the available command line options [For example, compare **man bison** and **info bison**.]

install-info Used to install info pages; it updates entries in the **info** index file

makeinfo Translates the given Texinfo source documents into info pages, plain text, or HTML

pdftexi2dvi Used to format the given Texinfo document into a Portable Document Format (PDF) file

pod2texi Converts Pod to Texinfo format

texi2any Translate Texinfo source documentation to various other formats

texi2dvi Used to format the given Texinfo document into a device-independent file that can be printed

texi2pdf Used to format the given Texinfo document into a Portable Document Format (PDF) file

texindex Used to sort Texinfo index files

6.71. Vim-8.1.1846

Vim - один из мощнейших текстовых редакторов с полной свободой настройки и автоматизации, возможными благодаря расширениям и надстройкам.

Приблизительное время сборки: 2.2 SBU

Требуемое дисковое пространство: 190 MB

Альтернативы текстовому редактору Vim

Если вы предпочитаете иной текстовый редактор, например Emacs,Joe или Nano, перейдите по ссылке *https://linuxfromscratch.ru/blfs/view/svn/postlfs/editors.html* чтобы ознакомиться со списком текстовых редакторов, доступных для установки. Перейдите в соответствующий раздел, и выполните инструкции по установки нужного пакета.

6.71.1. Установка пакета Vim

Для начала, изменим путь к файлу `vimrc` на каталог `/etc` :

```
echo '#define SYS_VIMRC_FILE "/etc/vimrc"' >> src/feature.h
```

Подготовьте пакет Vim к компиляции:

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

Скомпилируйте пакет:

```
make
```

Чтобы подготовить тесты к выполнению, убедитесь что пользователь nobody может выполнять запись в дерево исходных текстов пакета:

```
chown -Rv nobody .
```

Теперь запустите тесты от пользователяnobody:

```
su nobody -s /bin/bash -c "LANG=en_US.UTF-8 make -j1 test" &> vim-test.log
```

Набор тестов выводит на экран множество двоичных данных. Это может вызвать проблемы с настройками текущего терминала. Проблема может быть решена путем перенаправления вывода в файл журнала, как показано выше. успешный тест приведет к словам ALL DONE в файле журнала по окончании процесса тестирования.

Установите пакет:

```
make install
```

Многие пользователи используют команду **vi** вместо **vim**. Чтобы сохранить такую возможность, необходимо создать символическую ссылку для бинарного файла и для справочных руководств для предоставляемых языков:

```
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 к пакету устанавливается в каталог `/usr/share/vim` . Следующая символическая ссылка позволяет осуществить доступ к документации в каталоге `/usr/share/doc/vim-8.1.1846` , поскольку это согласуется с расположением документации для других пакетов:

```
ln -sv ../vim/vim81/doc /usr/share/doc/vim-8.1.1846
```

Если в дальнейшем планируется установка X Window System (конная система, обеспечивающая стандартные инструменты и протоколы для построения графического интерфейса пользователя. Используется в UNIX-подобных ОС) в системе LFS, то пакет потребуется повторно перекомпилировать, после установки X Window System. Vim поставляется с графическим интерфейсом который зависит от системы X и некоторых дополнительных библиотек. Для более полного ознакомления с процессом установки, изучите информацию в книге BLFS по следующей ссылке: *https://linuxfromscratch.ru/blfs/view/svn/postlfs/vim.html*.

6.71.2. Конфигурация Vim

По умолчанию, **vim** выполняется в vi-несовместимом режиме. Это может быть новым для пользователей, которые использовали другие редакторы ранее. Настройка“nocompatible” добавлена ниже, чтобы подчеркнуть тот факт, что будет использоваться новое поведение. Он также напоминает тем, кто хотел бы перейти в “совместимый” режим, что он должен быть

первым параметром в файле конфигурации. Это необходимо, поскольку он изменяет другие параметры и переопределения должны быть указаны только после этого параметра. 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
```

The *set nocompatible* setting makes **vim** behave in a more useful way (the default) than the vi-compatible manner. Remove the "no" to keep the old **vi** behavior. The *set backspace=2* setting allows backspacing over line breaks, autoindents, and the start of insert. The *syntax on* parameter enables vim's syntax highlighting. The *set mouse=* setting enables proper pasting of text with the mouse when working in chroot or over a remote connection. Finally, the *if* statement with the *set background=dark* setting corrects **vim**'s guess about the background color of some terminal emulators. This gives the highlighting a better color scheme for use on the black background of these programs.

Documentation for other available options can be obtained by running the следующую команду:

```
vim -c ':options'
```

Note

By default, Vim only installs spell files for the English language. To install spell files for your preferred language, download the `*.spl` and optionally, the `*.sug` files for your language and character encoding from *ftp://ftp.vim.org/pub/vim/runtime/spell/* and save them to `/usr/share/vim/vim81/spell/`.

To use these spell files, some configuration in `/etc/vimrc` is needed, e.g.:

```
set spelllang=en,ru
set spell
```

For more information, see the appropriate README file located at the URL above.

6.71.3. Содержимое пакета Vim

Установленные программы: ex (link to vim), rview (link to vim), rvim (link to vim), vi (link to vim), view (link to vim), vim, vimdiff (link to vim), vimtutor, and xxd
Каталог установки: /usr/share/vim

Краткое описание

ex Starts **vim** in ex mode

rview Is a restricted version of **view**; no shell commands can be started and **view** cannot be suspended

rvim Is a restricted version of **vim**; no shell commands can be started and **vim** cannot be suspended

vi Link to **vim**

view Starts **vim** in read-only mode

vim Is the editor

vimdiff Edits two or three versions of a file with **vim** and shows differences

vimtutor Teaches the basic keys and commands of **vim**

xxd Creates a hex dump of the given file; it can also do the reverse, so it can be used for binary patching

6.72. Procps-ng-3.3.15

Пакет Procps-ng содержит программы для мониторинга за процессами.

Приблизительное время сборки: 0.2 SBU

Требуемое дисковое пространство: 17 MB

6.72.1. Установка пакета Procps-ng

Подготовьте пакет procps-ng к компиляции:

```
./configure --prefix=/usr                            \
            --exec-prefix=                           \
            --libdir=/usr/lib                        \
            --docdir=/usr/share/doc/procps-ng-3.3.15 \
            --disable-static                         \
            --disable-kill
```

Значение параметров конфигурации:

`--disable-kill`
Этот аргумент отключает сборку программы **kill** которая будет содержится в пакете Util-linux и будет установлена позднее.

Скомпилируйте пакет:

```
make
```

Для выполнения набора тестов в среде LFS, требуется внести некоторые модификации. Удалите тесто которые не работает когда используется tty устройство и исправьте два других. Для выполнения тестов, выполните следующую команду:

```
sed -i -r 's|(pmap_initname)\\\$|\1|' testsuite/pmap.test/pmap.exp
sed -i '/set tty/d' testsuite/pkill.test/pkill.exp
rm testsuite/pgrep.test/pgrep.exp
make check
```

Установите пакет:

```
make install
```

Наконец, переместите необходимые библиотеки в каталог, который может быть найден, если каталог `/usr` не примонтирован.

```
mv -v /usr/lib/libprocps.so.* /lib
ln -sfv ../../lib/$(readlink /usr/lib/libprocps.so) /usr/lib/libprocps.so
```

6.72.2. Содержимое пакета Procps-ng

Установленные программы: free, pgrep, pidof, pkill, pmap, ps, pwdx, slabtop, sysctl, tload, top, uptime, vmstat, w, and watch
Установленная библиотека: libprocps.so
Установленные каталоги: /usr/include/proc and /usr/share/doc/procps-ng-3.3.15

Краткое описание

free Сообщает количество свободной и используемой памяти (как физических, так и файла подкачки) в системе

pgrep Поиск процессов на основе их имени и других атрибутов

pidof Сообщает PID указанной программы

pkill Обработка сигналов на основе их имени и других атрибутов

pmap Сообщает карту памяти данного процесса

ps Перечисляет текущие запущенные процессы

pwdx Сообщает текущий рабочий каталог процесса

slabtop Отображает подробную информацию о slab кеше в реальном времени

sysctl Изменяет параметры ядра во время выполнения

tload Распечатывает график текущей средней загрузки системы

top Отображает список наиболее интенсивных процессов; Это обеспечивает постоянный просмотр активности процессора в режиме реального времени

uptime Сообщает, сколько времени работает система, сколько пользователей вошедшие в систему и средние значения загрузки системы

vmstat Статистика виртуальной памяти, информация о процессах, памяти, пейджинге, блоке ввода / вывода (IO), ловушки и активность центрального процессора

w Показывает, какие пользователи в настоящее время зарегистрированы, где и с каких пор

watch Повторно запускает заданную команду, отображая первый полный экран его вывода; это позволяет пользователю наблюдать за изменениеми с течением времени

`libprocps` Содержит функции используемые многими проограммами пакета

6.73. Util-linux-2.34

Пакет содержит стандартный набор служебных утилит командной строки, такие как - утилиты для работы с файловой системой, консолью, разделами, и сообщениями.

Приблизительное время сборки: 1.2 SBU

Требуемое дисковое пространство: 283 MB

6.73.1. Замечания по поводу совместимости с FHS

Стандарт FHS рекомендует использовать каталог `/var/lib/hwclock` вместо каталога `/etc` где располагается файл `adjtime`. Создадим каталог чтобы включить хранилище программы **hwclock**:

```
mkdir -pv /var/lib/hwclock
```

6.73.2. Установка пакета Util-linux

Подготовьте пакет Util-linux к компиляции:

```
./configure ADJTIME_PATH=/var/lib/hwclock/adjtime   \
            --docdir=/usr/share/doc/util-linux-2.34 \
            --disable-chfn-chsh  \
            --disable-login      \
            --disable-nologin    \
            --disable-su         \
            --disable-setpriv    \
            --disable-runuser    \
            --disable-pylibmount \
            --disable-static     \
            --without-python     \
            --without-systemd    \
            --without-systemdsystemunitdir
```

Аргументы --disable и --without предотвратят появление предупреждений, о требуемых пакетах, которые не несовместимы с программами, установленными другими пакетами.

Скомпилируйте пакет:

```
make
```

По желанию, можно запустить тесты, от непревилегированного пользователя:

Warning

При запуске тестов от корневого пользователя можно повредить систему. Чтобы правильно запустить тесты, аргумент CONFIG_SCSI_DEBUG для текущего ядра должен быть доступен и должен быть доступен как модуль ядра. Сборка его в ядро помешает загрузке системы. Для полного покрытия тестами, необходимо установить некоторые пакеты из книги BLFS. По вашему желанию, тесты могут быть запущены после перезагрузки системы, выполнив команды:

```
bash tests/run.sh --srcdir=$PWD --builddir=$PWD
```

```
chown -Rv nobody .
su nobody -s /bin/bash -c "PATH=$PATH make -k check"
```

Установите пакет:

```
make install
```

6.73.3. 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, fdformat, fdisk, fincore, findfs, findmnt, flock, fsck, fsck.cramfs, fsck.minix, fsfreeze, fstrim, getopt, hexdump, hwclock, i386, ionice, ipcmk, ipcrm, ipcs, isosize, kill, last, lastb (link to last), ldattach, linux32, linux64, logger, look, losetup, lsblk, lscpu, lsipc, lslocks, lslogins, lsmem, lsns, mcookie, mesg, mkfs, mkfs.bfs, mkfs.cramfs, mkfs.minix, mkswap, more, mount, mountpoint, namei, nsenter, partx, pivot_root, prlimit, raw, readprofile, rename, renice, resizepart, rev, rfkill, rtcwake, script, scriptreplay, setarch, setsid, setterm, sfdisk, sulogin, swaplabel, swapoff (link to swapon), swapon, switch_root, taskset, ul, umount, uname26, unshare, utmpdump, uuidd, uuidgen, uuidparse, wall, wdctl, whereis, wipefs, x86_64, and zramctl
Установленные библиотеки:	libblkid.so, libfdisk.so, libmount.so, libsmartcols.so, and libuuid.so
Установленные каталоги:	/usr/include/blkid, /usr/include/libfdisk, /usr/include/libmount, /usr/include/libsmartcols, /usr/include/uuid, /usr/share/doc/util-linux-2.34, and /var/lib/hwclock

Краткое описание

addpart	Informs the Linux kernel of new partitions
agetty	Opens a tty port, prompts for a login name, and then invokes the **login** program
blkdiscard	Discards sectors on a device
blkid	A command line utility to locate and print block device attributes
blkzone	Runs zone command on the given block device
blockdev	Allows users to call block device ioctls from the command line
cal	Displays a simple calendar

cfdisk	Manipulates the partition table of the given device
chcpu	Modifies the state of CPUs
chmem	Configures memory
choom	Отображает и настраивает показатель OOM-killer
chrt	Manipulates real-time attributes of a process
col	Filters out reverse line feeds
colcrt	Filters **nroff** output for terminals that lack some capabilities, such as overstriking and half-lines
colrm	Filters out the given columns
column	Formats a given file into multiple columns
ctrlaltdel	Sets the function of the Ctrl+Alt+Del key combination to a hard or a soft reset
delpart	Asks the Linux kernel to remove a partition
dmesg	Dumps the kernel boot messages
eject	Ejects removable media
fallocate	Preallocates space to a file
fdformat	Low-level formats a floppy disk
fdisk	Manipulates the partition table of the given device
fincore	Counts pages of file contents in core
findfs	Finds a file system by label or Universally Unique Identifier (UUID)
findmnt	Is a command line interface to the libmount library for work with mountinfo, fstab and mtab files
flock	Acquires a file lock and then executes a command with the lock held
fsck	Is used to check, and optionally repair, file systems
fsck.cramfs	Performs a consistency check on the Cramfs file system on the given device
fsck.minix	Performs a consistency check on the Minix file system on the given device
fsfreeze	Is a very simple wrapper around FIFREEZE/FITHAW ioctl kernel driver operations
fstrim	Discards unused blocks on a mounted filesystem
getopt	Parses options in the given command line
hexdump	Dumps the given file in hexadecimal or in another given format
hwclock	Reads or sets the system's hardware clock, also called the Real-Time Clock (RTC) or Basic Input-Output System (BIOS) clock
i386	A symbolic link to setarch
ionice	Gets or sets the io scheduling class and priority for a program
ipcmk	Creates various IPC resources
ipcrm	Removes the given Inter-Process Communication (IPC) resource
ipcs	Provides IPC status information
isosize	Reports the size of an iso9660 file system

kill Sends signals to processes

last Shows which users last logged in (and out), searching back through the `/var/log/wtmp` file; it also shows system boots, shutdowns, и выполните команду-level changes

lastb Shows the failed login attempts, as logged in `/var/log/btmp`

ldattach Attaches a line discipline to a serial line

linux32 A symbolic link to setarch

linux64 A symbolic link to setarch

logger Enters the given message into the system log

look Displays lines that begin with the given string

losetup Sets up and controls loop devices

lsblk Lists information about all or selected block devices in a tree-like format

lscpu Prints CPU architecture information

lsipc Prints information on IPC facilities currently employed in the system

lslocks Lists local system locks

lslogins Lists information about users, groups and system accounts

lsmem Lists the ranges of available memory with their online status

lsns Lists namespaces

mcookie Generates magic cookies (128-bit random hexadecimal numbers) for **xauth**

mesg Controls whether other users can send messages to the current user's terminal

mkfs Builds a file system on a device (usually a hard disk partition)

mkfs.bfs Creates a Santa Cruz Operations (SCO) bfs file system

mkfs.cramfs Creates a cramfs file system

mkfs.minix Creates a Minix file system

mkswap Initializes the given device or file to be used as a swap area

more A filter for paging through text one screen at a time

mount Attaches the file system on the given device to a specified directory in the file-system tree

mountpoint Checks if the directory is a mountpoint

namei Shows the symbolic links in the given pathnames

nsenter Runs a program with namespaces of other processes

partx Tells the kernel about the presence and numbering of on-disk partitions

pivot_root Makes the given file system the new root file system of the current process

prlimit Get and set a process' resource limits

raw Bind a Linux raw character device to a block device

readprofile Reads kernel profiling information

rename Renames the given files, replacing a given string with another

renice	Alters the priority of running processes
resizepart	Asks the Linux kernel to resize a partition
rev	Reverses the lines of a given file
rkfill	Tool for enabling and disabling wireless devices
rtcwake	Used to enter a system sleep state until specified wakeup time
script	Makes a typescript of a terminal session
scriptreplay	Plays back typescripts using timing information
setarch	Changes reported architecture in a new program environment and sets personality flags
setsid	Runs the given program in a new session
setterm	Sets terminal attributes
sfdisk	A disk partition table manipulator
sulogin	Allows `root` to log in; it is normally invoked by **init** when the system goes into single user mode
swaplabel	Allows to change swaparea UUID and label
swapoff	Disables devices and files for paging and swapping
swapon	Enables devices and files for paging and swapping and lists the devices and files currently in use
switch_root	Switches to another filesystem as the root of the mount tree
tailf	Tracks the growth of a log file; displays the last 10 lines of a log file, then continues displaying any new entries in the log file as they are created
taskset	Retrieves or sets a process' CPU affinity
ul	A filter for translating underscores into escape sequences indicating underlining for the terminal in use
umount	Disconnects a file system from the system's file tree
uname26	A symbolic link to setarch
unshare	Runs a program with some namespaces unshared from parent
utmpdump	Displays the content of the given login file in a more user-friendly format
uuidd	A daemon used by the UUID library to generate time-based UUIDs in a secure and guaranteed-unique fashion
uuidgen	Creates new UUIDs. Each new UUID can reasonably be considered unique among all UUIDs created, on the local system and on other systems, in the past and in the future
uuidparse	An utility to parse unique identifiers
wall	Displays the contents of a file or, by default, its standard input, on the terminals of all currently logged in users
wdctl	Shows hardware watchdog status
whereis	Reports the location of the binary, source, and man page for the given command
wipefs	Wipes a filesystem signature from a device

x86_64	A symbolic link to setarch
zramctl	A program to set up and control zram (compressed ram disk) devices
`libblkid`	Содержит routines for device identification and token extraction
`libfdisk`	Содержит routines for manipulating partition tables
`libmount`	Содержит routines for block device mounting and unmounting
`libsmartcols`	Содержит routines for aiding screen output in tabular form
`libuuid`	Содержит routines for generating unique identifiers for objects that may be accessible beyond the local system

6.74. E2fsprogs-1.45.3

Утилиты для работы с файловыми системами ext2, ext3 и ext4. Это наиболее распространенные и тщательно протестированные файловые системы, поддерживаемые в Linux.

Приблизительное время сборки: 3.1 SBU
Требуемое дисковое пространство: 108 MB

6.74.1. Установка пакета E2fsprogs

В документации к E2fsprogs указано, что следует выполнять сборку в отдельном каталоге:

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

Подготовьте пакет E2fsprogs к компиляции:

```
../configure --prefix=/usr           \
             --bindir=/bin           \
             --with-root-prefix=""   \
             --enable-elf-shlibs     \
             --disable-libblkid      \
             --disable-libuuid       \
             --disable-uuidd         \
             --disable-fsck
```

Значения переменных среды и аргументов конфигурации:

`--with-root-prefix=""` и *`--bindir=/bin`*
Некоторые программы (такие как **e2fsck**) считаются важными программами. Когда, например, каталог `/usr` не примонтирован, необходимо, чтобы эти программы оставались доступны. Они принадлежат каталогам `/lib` и `/sbin` . Если эти аргументы не указать, установка будет произведена в каталог `/usr` .

`--enable-elf-shlibs`
Это создает общие библиотеки, которые используются некоторыми программами пакета.

`--disable-*`
Исключает установку библиотек `libuuid` и `libblkid`, службы `uuidd`,и **fsck** обертки, которые содержатся в пакете Util-Linux и являются более актуальными.

Скомпилируйте пакет:

```
make
```

Для выполнения тестов, выполните команду:

```
make check
```

Один из тестов пакета E2fsprogs попытается выделить 256 MB памяти. Если у вас нет такого объема оперативной памяти, не забудьте включить требуемое пространство в файл подкачки для выполнения теста. Смотрите Section 2.5, “Создание файловой системы на разделе” и

Section 2.7, “Монтирование нового раздела” чтобы ознакомиться с информацией по поводу создания и включения файла подкчки. Два теста, f_bigalloc_badinode and f_bigalloc_orphan_list, известно что не пройдут.

Установите бинарные файлы, документацию, и разделяемые библиотеки:

```
make install
```

Установите статические библиотеки и заголовочные файлы:

```
make install-libs
```

Установите права на запись статическим библиотекам, для того, чтобы позднее, удалить отладочные символы:

```
chmod -v u+w /usr/lib/{libcom_err,libe2p,libext2fs,libss}.a
```

Пакет установит запакованный файл `.info` и не обновит системный файл `dir`. Распакуйте файл и обновите файл `dir`, выполнив следующую команду:

```
gunzip -v /usr/share/info/libext2fs.info.gz
install-info --dir-file=/usr/share/info/dir /usr/share/info/libext2fs.info
```

При необходимости создайте и установите дополнительную документацию выполнение 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
```

6.74.2. 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, and tune2fs
Установленные библиотеки:	libcom_err.so, libe2p.so, libext2fs.so, and libss.so
Установленные каталоги:	/usr/include/e2p, /usr/include/et, /usr/include/ext2fs, /usr/include/ss, /usr/lib/e2fsprogs, /usr/share/et, and /usr/share/ss

Краткое описание

badblocks	Поиск поврежденных блоков на устройстве (обычно дисковый раздел)
chattr	Изменение атрибутов файлов на файловой системе `ext2`; также изменяет файловую систему `ext3` , журналирование файловой системы `ext2`
compile_et	Компилятор таблицы ошибок; он преобразует таблицу кода ошибки имена и сообщения в исходный файл C, подходящий для использования с библиотекой `com_err`
debugfs	Отладчик файловой системы; A file system debugger; его можно использовать для изучения и изменения состояния файловой системы `ext2`

dumpe2fs	Распечатывает супер-блок и блокирует информацию о группе для файловой системы, присутствующей на устройстве
e2freefrag	Сообщает информацию о фрагментации свободного пространства
e2fsck	Используется для проверки и, при необходимости, ремонта файловых системext2 и ext3
e2image	Используется для сохранения критичных данных файловой системы ext2 в файл
e2label	Отображает или изменяет метку файловой системы для файловой системы ext2на данном устройстве
e2mmpstatus	Checks MMP status of an ext4 filesystem
e2scrub	Checks the contents of a mounted ext[234] filesystem
e2scrub_all	Checks all mounted ext[234] filesystems for errors
e2undo	Заменяет журнал undo_log для файловых систем ext2/ext3/ext4 на данном устройстве [можно использовать для отмены неудачной операции при использовании программы e2fsprogs.]
e4crypt	Ext4 filesystem encryption utility
e4defrag	Онлайн дефрагментатор для файловой системы ext4
filefrag	Отчет о том, насколько сильно файл может быть фрагментированным
fsck.ext2	Проверка файловой системы ext2 и жесткая ссылка на **e2fsck**
fsck.ext3	Проверка файловой системыext3 и жесткая ссылка на **e2fsck**
fsck.ext4	Проверка файловой системы ext4 и жесткая ссылка на **e2fsck**
logsave	Сохраняет вывод команды в файле журнала
lsattr	Перечисляет атрибуты файлов во второй расширенной файловой системе
mk_cmds	Преобразует таблицу имен команд и справочных сообщений в C исходный файл, подходящий для использования с библиотекой подсистемыlibss
mke2fs	Создает файловую систему ext2 или ext3 на устройстве
mkfs.ext2	Создает файловую систему ext2 и является жесткой ссылкой на **mke2fs**
mkfs.ext3	Создает файловую систему ext3 и является жесткой ссылкой на **mke2fs**
mkfs.ext4	Создает файловую систему ext4 и является жесткой ссылкой на **mke2fs**
mklost+found	Создает каталог lost+found на файловой системе ext2; он предварительно выделяет блоки диска для каталога, для выполнения задач **e2fsck**
resize2fs	Может использоваться для увеличения или сжатия файловой системыext2
tune2fs	Используется для модифицирования параметров файловой системы ext2
libcom_err	Общая процедура отображения ошибок
libe2p	Используется **dumpe2fs**, **chattr**, и **lsattr**
libext2fs	Содержит программы позволяющие программам пользовательского уровня манипулировать файловыми системами ext2
libss	Используется **debugfs**

6.75. Sysklogd-1.5.1

Системные утилиты, которые обеспечивают поддержку журналирования системы и перехват сообщений ядра. Поддержка доменных гнёзд интернет и unix позволяет этому пакету утилит поддерживать и локальное и удалённое журналирование.

Приблизительное время сборки: less than 0.1 SBU
Требуемое дисковое пространство: 0.6 MB

6.75.1. Установка пакета Sysklogd

Во-первых, устраните проблемы, которые приводят к ошибке сегментации - Segmentation fault (ошибка программного обеспечения, возникающая при попытке обращения к недоступным для записи участкам памяти либо при попытке изменить память запрещённым способом.) и исправьте некоторые условия в klogd для устаревших конструкций программ:

```
sed -i '/Error loading kernel symbols/{n;n;d}' ksym_mod.c
sed -i 's/union wait/int/' syslogd.c
```

Скомпилируйте пакет:

```
make
```

У этого пакета нет тестов.

Установите пакет:

```
make BINDIR=/sbin install
```

6.75.2. 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 *

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

6.75.3. Содержимое пакета Sysklogd

Установленные программы: klogd and syslogd

Краткое описание

klogd Системная служба для перехвата и регистрации сообщений ядра

syslogd Регистрирует сообщения, предлагаемые системными программами для ведения журнала [Каждое зарегистрированное сообщение содержит как минимум отметку даты и имя хост-системы, и обычно это имя программы, но это зависит от доверия служб регистрации.]

6.76. Sysvinit-2.95

Пакет Sysvinit содержит программы для управления загрузкой, выполнением, и выключением системы.

Приблизительное время сборки: less than 0.1 SBU
Требуемое дисковое пространство: 1.3 MB

6.76.1. Установка пакета Sysvinit

Во-первых, примените патч, который удаляет несколько программ, установленных другими пакетами, очищает сообщение и исправляет предупреждение компилятора:

```
patch -Np1 -i ../sysvinit-2.95-consolidated-1.patch
```

Скомпилируйте пакет:

```
make
```

У этого пакета нет тестов.

Установите пакет:

```
make install
```

6.76.2. Содержимое пакета Sysvinit

Установленные программы: bootlogd, fstab-decode, halt, init, killall5, poweroff (link to halt), reboot (link to halt), runlevel, shutdown, and telinit (link to init)

Краткое описание

bootlogd Logs boot messages to a log file

fstab-decode Run a command with fstab-encoded arguments

halt Normally invokes **shutdown** with the *-h* option, except when already in run-level 0, then it tells the kernel to halt the system; it notes in the file `/var/log/wtmp` that the system is being brought down

init The first process to be started when the kernel has initialized the hardware which takes over the boot process and starts all the proceses specified in its configuration file

killall5 Sends a signal to all processes, except the processes in its own session so it will not kill its parent shell

poweroff Tells the kernel to halt the system and switch off the computer (see **halt**)

reboot Tells the kernel to reboot the system (see **halt**)

runlevel Reports the previous and the current run-level, as noted in the last run-level record in `/var/run/utmp`

shutdown Brings the system down in a secure way, signaling all processes and notifying all logged-in users

telinit Tells **init** which run-level to change to

6.77. Eudev-3.2.8

Eudev - диспетчер устройств. Он контролирует записи в каталоге /dev, так как устройства добавляются или удаляются из системы динамически.

Приблизительное время сборки: 0.2 SBU
Требуемое дисковое пространство: 82 MB

6.77.1. Установка пакета Eudev

Подготовьте пакет Eudev к компиляции:

```
./configure --prefix=/usr           \
            --bindir=/sbin          \
            --sbindir=/sbin         \
            --libdir=/usr/lib       \
            --sysconfdir=/etc       \
            --libexecdir=/lib       \
            --with-rootprefix=      \
            --with-rootlibdir=/lib  \
            --enable-manpages       \
            --disable-static
```

Скомпилируйте пакет:

Создайте каталоги, которые необходимы для успешного прохождения тестов, которые также будут использоваться в процессе установки:

```
mkdir -pv /lib/udev/rules.d
mkdir -pv /etc/udev/rules.d
```

Для выполнения тестов, выполните команду:

```
make check
```

Установите пакет:

```
make install
```

Установите дополнительные правила и файлы, используемые в окружении LFS:

```
tar -xvf ../udev-lfs-20171102.tar.xz
make -f udev-lfs-20171102/Makefile.lfs install
```

6.77.2. Настройка Eudev

Информация об устройствах располагается в файлах `/etc/udev/hwdb.d` и каталогах `/lib/udev/hwdb.d` . Eudev требуется чтобы эта информация была скомпилирована в двоичную базу данных в каталоге `/etc/udev/hwdb.bin` . Создадим пустую базу данных:

```
udevadm hwdb --update
```

Эта команда должна быть запущена каждый раз, когда информация об оборудовании обновится.

6.77.3. Содержимое пакета Eudev

Установленные программы:	udevadm and udevd
Установленные библиотеки:	libudev.so
Установленные каталоги:	/etc/udev, /lib/udev, and /usr/share/doc/udev-udev-lfs-20171102

Краткое описание

udevadm	Generic udev administration tool: controls the udevd daemon, provides info from the Udev database, monitors uevents, waits for uevents to finish, tests Udev configuration, and triggers uevents for a given device
udevd	A daemon that listens for uevents on the netlink socket, creates devices и выполните командys the configured external programs in response to these uevents
`libudev`	A library interface to udev device information
`/etc/udev`	Содержит Udev configuration files, device permissions, and rules for device naming

6.78. По поводу отладочных символов

Большинство программ и библиотек по умолчанию компилируются с отладочными символами (с указанием аргумента *-g* команде **gcc**). Это означает, что при проведении отладки программы или библиотеки, которая была скомпилирована с включённой отладочной информацией, отладчик может предоставить информацию не только по адресам памяти, но также будет предоставлена информация об именах функций и переменных.

Однако, включение отладочных символов значительно увеличивает размер программы или библиотеки. Ниже приведена информация по объему пространства, занимаемого отладочными символами:

- **bash** двоичный файл с отладочными символами: 1200 KB
- **bash** двоичный файл без отладочных символов: 480 KB
- Файлы Glibc и GCC (`/lib` and `/usr/lib`) с отладочными символами: 87 MB
- Файлы Glibc и GCC без отладочных символов: 16 MB

Размеры могут варьироваться в зависимости от того, какой компилятор и библиотека Си были использованы. Но сравнивая программы с и без отладочных символов, разница как правило будет от двух до пяти раз.

Поскольку большинство пользователей никогда не будет использовать отладчик для системного программного обеспечения, можно восполнить достаточный объем дискового пространства, удалив отладочные символы. В следующем в разделе рассказано, как удалить все отладочные символы из программ и библиотек.

6.79. Повторная очистка от отладочных символов

Этот раздел является необязательным. Если вы не программист и не планируете выполнять отладку установленного программного обеспечения, то общий размер системы можно сократить примерно на 90 МБ, удалив отладочные символы из двоичных файлов и библиотек. Такая операция не вызовет проблем, за исключением того, что не получится выполнить отладку установленного программного обеспечения.

Большинство пользователей, которые используют приведенные ниже команды, не испытывают никаких трудностей. Тем не менее, легко сделать опечатку и сделать новую систему непригодной для использования, поэтому перед запуском команд рекомендуется создать резервную копию системы LFS в текущем состоянии.

Сначала расположим отладочные символы для выбранных библиотек в отдельные файлы. Отладочная информация необходима для запуска регрессионных тестов, которые используют *valgrind* или *gdb* позднее в книге BLFS.

```
save_lib="ld-2.30.so libc-2.30.so libpthread-2.30.so libthread_db-1.0.so"

cd /lib

for LIB in $save_lib; do
    objcopy --only-keep-debug $LIB $LIB.dbg
    strip --strip-unneeded $LIB
    objcopy --add-gnu-debuglink=$LIB.dbg $LIB
done

save_usrlib="libquadmath.so.0.0.0 libstdc++.so.6.0.27
             libitm.so.1.0.0 libatomic.so.1.2.0"

cd /usr/lib

for LIB in $save_usrlib; do
    objcopy --only-keep-debug $LIB $LIB.dbg
    strip --strip-unneeded $LIB
    objcopy --add-gnu-debuglink=$LIB.dbg $LIB
done

unset LIB save_lib save_usrlib
```

Перед выполнением очистки тщательно следите за тем, чтобы ни один из исполняемых файлов, который планируется удалить, не был запущен. Если вы не уверены, что вы находитесь в правильно настроенной среде chroot как было рассказано ранее, Section 6.4, “Вход в окружение Chroot,” сперва выйдем из среды chroot, выполнив команду:

```
logout
```

Выполним вход в среду chroot заново, выполнив команду:

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

Теперь двоичные файлы и библиотеки можно безопасно очистить:

```
/tools/bin/find /usr/lib -type f -name \*.a \
   -exec /tools/bin/strip --strip-debug {} ';'

/tools/bin/find /lib /usr/lib -type f \( -name \*.so* -a ! -name \*dbg \) \
   -exec /tools/bin/strip --strip-unneeded {} ';'

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

На экране будет множество сообщений, не удается распознать формат файлов (format not recognized). Такие предупреждения можно игнорировать. Они указывают, что файлы являются сценариями вместо двоичных файлов.

6.80. Выполнение очистки

Наконец, очистите некоторые дополнительные файлы, оставшиеся от запускаемых ранее тестов:

```
rm -rf /tmp/*
```

С этого момента, когда вы возвращаетесь в среду chroot после выхода, необходимо использовать следующую измененную команду входа в chroot окружение:

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

Причина этому является то, что программы в каталоге `/tools` больше не потребуются, и соответствующий каталог `/tools` , при желании, можно удалить.

Note

При удалении каталога `/tools` также, будут удалены временные копии программ Tcl, Expect, и DejaGNU, которые были установлены для обеспечения запуска тестов устанавливаемых пакетов. Если эти программы вам вновь понадобятся, вы можете установить их позднее. В книге BLFS есть информация по установке и настройке этих пакетов (смотрите по ссылке *https://linuxfromscratch.ru/blfs/*).

Если виртуальные файловые системы не были примонтированы, по причине перезагрузки или ручного отмонтирования,убедитесь что они заново примонтированы, когда вы входите в среду chroot. Этот процесс описан по ссылке Section 6.2.2, “Монтирование и заполнение каталога /dev” и Section 6.2.3, “Монтирование виртуальных файловых систем ядра”.

Есть статические библиотеки, которые ранее не были удалены в этой главе, чтобы регрессионные тесты могли правильно выполняться, в нескольких пакетах. Эти библиотеки из binutils, bzip2, e2fsprogs, flex, libtool и zlib, при желании, можно тоже удалить:

```
rm -f /usr/lib/lib{bfd,opcodes}.a
rm -f /usr/lib/libbz2.a
rm -f /usr/lib/lib{com_err,e2p,ext2fs,ss}.a
rm -f /usr/lib/libltdl.a
rm -f /usr/lib/libfl.a
rm -f /usr/lib/libfl_pic.a
rm -f /usr/lib/libz.a
```

Есть еще несколько файлов с расширением .la, расположенные каталогах в /usr/lib и /usr/libexec. Это файлы "архивов libtool", и в системе linux они не нужны. На данный момент они не требуется. Чтобы удалить их, выполните команду:

```
find /usr/lib /usr/libexec -name \*.la -delete
```

Более подробную информацию вы можете изучить в книге BLFS, перейдя по ссылке *в разделе "Информация об архивных файлах (.la)"*.

Chapter 7. Конфигурация системы

7.1. Введение

Загрузка Linux системы включает в себя несколько задач. Процесс сперва должен примонтировать как виртуальные, так и реальные файловые системы, инициализировать устройства, активировать файл подкачки, проверить целостность файловых систем, примонтировать другие разделы или файл подкачки, установить системные часы, создать сети, запустить службы, требуемые системой и выполнить любые другие задачи, необходимые пользователю. Этот процесс должен быть организован для выполнения задач в правильном порядке и, в то же время, выполняться как можно быстрее.

7.1.1. System V

System V - представляет классический процесс загрузки, который использовался в Unix и Unix-подобных системах, таких как Linux с 1983 года. Он состоит из небольшой программы **init**, которая устанавливает базовые программы, такие как **login** (через getty) и запускает другие сценарии. Этот сценарий, как правило, называется rc, и управляет выполнением дополнительных сценариев, необходимых для инициализации системы.

Программа init управляется файлом `/etc/inittab` и структурирована на уровни запуска, которые могут выполнятся пользователем:

```
0 — Halt - остановка
1 — Single user mode - Однопользовательский режим
2 — Multiuser, without networking - Многопользовательский режим, без сети
3 — Full multiuser mode - Полный многопользовательский режим
4 — User definable - Режим, определяемый пользователем
5 — Full multiuser mode with display manager - Полный многопользовательский режим с экранным м
6 — reboot - перезагрузка
```

Обычно, стандартный уровень запуска - 3 или 5.

Преимущества

- Устоявшаяся, хорошо понятая система
- Легко настроить.

Недостатки

- Медленная загрузка. Средняя скорость загрузки LFS системы составляет 8-12 секунд, если измерить время с начала первого отображения приглашения командной строки ядра. Соединение с сетью как правило устанавливается примерно 2 секунды после приглашения командной строки.
- Последовательная обработка задач загрузки. Это связано с предыдущим пунктом. Задержка в работе любого процесса, например проверка файловой системы, задержит соответственно и весь процесс загрузки.

- Не поддерживает напрямую дополнительные функции, такие как контроль групп (cgroups) и расписания для каждого пользователя.
- Для добавления сценариев требуются ручные, статические последовательные решения.

7.2. LFS-Bootscripts-20190524

Пакет LFS-Bootscripts содержит набор сценариев для запуска/остановки LFS системы. Файлы конфигурации и процедуры необходимы, чтобы настроить процесс загрузки, писанные в следующих разделах.

Приблизительное время сборки: less than 0.1 SBU
Требуемое дисковое пространство: 244 KB

7.2.1. Установка пакета LFS-Bootscripts

Установите пакет:

```
make install
```

7.2.2. Содержимое пакета LFS-Bootscripts

Установленные сценарии: checkfs, cleanfs, console, functions, halt, ifdown, ifup, localnet, modules, mountfs, mountvirtfs, network, rc, reboot, sendsignals, setclock, ipv4-static, swap, sysctl, sysklogd, template, udev, and udev_retry
Установленные каталоги: /etc/rc.d, /etc/init.d (symbolic link), /etc/sysconfig, /lib/services, /lib/lsb (symbolic link)

Краткое описание

checkfs Проверяет целостность файловых систем, после того, как они были примонтированы (за исключением журналов и сетевых файловых систем)

cleanfs Удаляет файлы, которые не должны быть сохранены между циклами включения/выключения системы, такие как в каталоге `/var/run/` и `/var/lock/` ; также создается заново `/var/run/utmp` и удаляют, возможные наличия файлов `/etc/nologin` , `/fastboot` , и `/forcefsck`

console Загружает правильные таблицы раскладки клавиатуры для клавиатуры, и устанавливает шрифт экрана

functions Содержит общие функции, такие как error и проверка статуса. Эти функции используются другими сценариями пакета

halt Останавливает систему

ifdown Остановка работы сетевых устройств

ifup Инициализирует сетевые устройства

localnet Выполняет настройку имени хоста и локального loopback устройства

modules Загружает модули ядра из списка, который находится в файле `/etc/sysconfig/modules` , используя переданные здесь аргументы

mountfs Монтирует все файловые системы, кроме тех, которые отмечены как *noauto* или сетевые файловые системы

mountvirtfs Монтирует виртуальные файловые системы, такие как `proc`

network Выполняет настройку сетевых интерфейсов, таких как сетевые карты, и настаивает сетевой шлюз по-умолчанию (где это применимо)

rc Основной управляющий сценарий. Он отвечает за запуск других сценариев загрузки один за одним, в той последовательности, которая определенной по имени обрабатываемых символических ссылок.

reboot Перезагружает систему

sendsignals Гарантирует то, что каждый процесс будет завершен до того, как система будет остановлена или перезагружена

setclock Сбрасывает настроенные часы ядра в локальное время в случае, если аппаратное время не настроено в UTC

ipv4-static Предоставляет функциональность, позволяющую назначать статический IP адрес сетевому интерфейсу

swap Включает или выключает файл подкачки и связанные с ним разделы файловой системы

sysctl Загружает параметры системной конфигурации в запущенное ядро из файла `/etc/sysctl.conf`, в том случае, если файл существует

sysklogd Запускает или останавливает системные службы журналирования и службы ядра

template Шаблон, для создания собственных сценариев загрузки для других служб

udev Подготовка каталога `/dev` и запуск Udev

udev_retry Повторная попытка запуска неудавшихся событий udev, копирование сгенерированных правил из каталога `/run/udev` в каталог `/etc/udev/rules.d` при необходимости

7.3. Обработка устройств и модулей

Ранее, в шестой главе, был установлен пакет eudev . Перед тем, как приступить к описанию процесса как это работает, будет кратко рассказана история предыдущих методов работы устройств.

Системы Linux традиционно используют метод статического создания устройств, посредством чего, большое число узлов устройств (иногда буквально тысячи узлов) создается в каталоге /dev , независимо от того, существуют ли соответствующие аппаратные устройства. Это обычно делается средствами сценария **MAKEDEV**, который содержит команды вызова программы **mknod** с соответствующим числом для каждого возможного устройства которое только может существовать в мире.

Используя метод Udev, только те устройства, которые были обнаружены ядром, получают свой узел. Поскольку эти узлы будут создаваться каждый раз, при загрузке системы, они будут располагаться в каталоге виртуальной файловой системы devtmpfs (виртуальная файловая система, которая полностью находится в оперативной памяти). Узлы не занимают много места в памяти, и их общий размер незначителен.

7.3.1. 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.

Главная проблема в этом подходе заключалась в механизме обнаружения, создания и назначения имен устройствам. Последняя из которых, связанная с назначением имен узлам устройств была, самой важной. Как правило, если имена устройствам можно настраивать, то политика назначения имён должна быть установлена системным администратором и не должна быть навязана разработчиками. Файловая система devfs также страдала от условий гонки, которые были присущи ее дизайну и не могли быть исправлены без существенной переработки самого ядра. В конечном счёте, эта файловая система была отмечена как устаревшая, на протяжении достаточно долгого периода, по причине отсутствия её ненадлежащей поддержки, и была удалена из ветки ядра в июне 2006 года.

При развитии нестабильной ветки ядра 2.5, позднее, выпущенной как стабильный релиз 2.6, появилась новая виртуальная файловая система sysfs. Задача этой файловой системы заключалась в экспорте представления об аппаратной конфигурации системы в процессах пользовательского окружения. Благодаря этому, разработка замены пользовательского окружения для devfs стала гораздо реалистичнее.

7.3.2. Реализация Udev

7.3.2.1. 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, напрямую регистрируют объекты с помощью sysfs (внутри devtmpfs), так как они обнаруживаются ядром. Для драйверов, которые скомпилированы в виде модулей, регистрация будет

происходить при его загрузке. После того, как файловая система sysfs будет примонтирована в каталог /sys , данные, которые регистрируются драйверами, с помощью sysfs, станут доступны для процессов пользовательского окружения и для udevd, для последующей обработки (включая изменения узлов устройств).

7.3.2.2. Создание узлов устройств

Файлы устройств создаются ядром при помощи файловой системы devtmpfs. Любой драйвер, которому необходимо зарегистрировать узел устройства, будет проходить через файловую систему devtmpfs (через системный драйвер ядра). Когда экземпляр devtmpfs монтируется в каталог /dev , узел устройства будет создан с фиксированным наименованием, соответствующими 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 сменит запись (наименование) во внутренней базе данных udevd для этого объекта.

Правила в этих трёх каталогах нумеруются и объединяются вместе. Если **udevd** не может найти правило для устройства, он оставит права доступа и права собственности на все первоначально используемые devtmpfs.

7.3.2.3. Загрузка модулей

Драйверы устройств, скомпилированные в виде модулей ядра могут содержать встроенные псевдонимы. Псевдонимы можно увидеть просмотрев вывод программы **modinfo**. Они, как правило, связаны с идентификаторам шины устройства, поддерживаемым 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 (которая должна совпадать с содержимым файла modalias в sysfs), тем самым загружая все модули, чьи псевдонимы совпадают в строке после расширение подстановочных знаков.

К указанному выше примеру, это означает, что в дополнение к snd-fm801 устаревший (и нежелательный) драйвер forte будет загружен, если он будет доступен. Ниже описано, как можно предотвратить загрузку нежелательных драйверов.

Само ядро также может загружать модули для сетевых протоколов, файловых систем и поддержки NLS по требованию.

7.3.2.4. Обработка устройств с горячей заменой или динамических устройств

При подключении устройства, например, MP3-плеер, к универсальной последовательной шине (USB), ядро распознает, что устройство подключено, и сгенерирует событие uevent. Это событие затем обрабатывается **udevd**, как было описано выше.

7.3.3. Проблемы с загрузкой модулей и созданием устройств

Есть несколько возможных проблем, когда дело доходит до автоматического создания узлов устройств.

7.3.3.1. Модуль ядра не загружается автоматически

Udev будет загружать модуль, только если он имеет указанный псевдоним шины, и драйвер шины правильно экспортирует необходимые псевдонимы в sysfs. В других случаях следует организовать загрузку модуля другими способами. начиная с версии Linux-5.2.8, в udev, как известно,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 общаться с ним 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-5.2.8, это относится к шинам 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 драйверов, не относящихся к оборудованию, например *loop*

7.3.3.2. Модуль ядра не загружается автоматически, и Udev не предназначен для его загрузки

Если модуль “обёртка” только расширяет функциональность, которая может быть предоставлена иным 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:`. Обратитесь к документации `modprobe.d(5)` для изучения синтаксиса и возможностей “softdep”.

Если модуль не является обёрткой, и полезен сам по себе, настройте загрузочный сценарий модулей, чтобы он добавлялся при загрузке системы. Для этого добавьте имя модуля в файл `/etc/sysconfig/modules` в отдельной строке. Этот способ сработает и для оберточных модулей, но не является оптимальным.

7.3.3.3. Udev загружает ненужные модули

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
```

Блокированные модули могут быть загружены вручную с явным указанием в команде **modprobe**.

7.3.3.4. Udev создает устройство неправильно или создает некорректную символическую ссылку

Это обычно происходит, если правило неожиданно совпадает с устройством. Например, плохо написанное правило может соответствовать как диску SCSI (по желанию), так и соответствующему универсальному устройству SCSI (неправильно) указанному поставщиком. Найдите нарушающее правило и уточните его с помощью команды **udevadm info**.

7.3.3.5. 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 (создайте его, если он не существует). Пожалуйста, оповестите в списке рассылки разработчиков LFS, если вы делаете это, и это помогает.

7.3.3.6. Udev не создаёт устройство

Предполагается, что драйвер статически встроен в ядро или уже загружен как модуль, и что вы уже проверили, что Udev не создает устройство с неправильным наименованием.

Udev не обладает информацией, необходимой для создания узла устройства, если драйвер ядра не экспортирует свои данные в `sysfs`. Как правило, такое происходит с внешними драйверами, которых нет в дереве исходного кода ядра. Создайте статический узел в каталоге `/lib/udev/devices` с соответствующими первичными и второстепенными номерами (см. файл devices.txt внутри документации ядра или документации, предоставленной сторонним поставщиком драйвера). Статический узел будет скопирован в `/dev` с помощью **udev**.

7.3.3.7. Порядок именования устройств изменяется случайным образом после перезагрузки

Это связано с тем, что Udev обрабатывает события и загружает модули параллельно, а значит в непредсказуемом порядке. Это никогда не будет “исправлено”. Не следует полагаться на стабильность имен устройств ядра. Вместо этого создайте свои собственные правила, которые делают символические ссылки со стабильными именами на основе некоторых стабильных атрибутов устройства, таких как серийный номер или вывод различных утилит *_id, установленных Udev. В разделе Section 7.4, “Управление устройствами” и Section 7.5, “Конфигурация Сети”, есть примеры.

7.3.4. Полезно для ознакомления

Дополнительная полезная документация доступна по следующим ссылкам:

- Реализация `devfs` в пользовательском окружении *http://www.kroah.com/linux/talks/ols_2003_udev_paper/Reprint-Kroah-Hartman-OLS2003.pdf*
- Файловая система `sysfs` *http://www.kernel.org/pub/linux/kernel/people/mochel/doc/papers/ols-2005/mochel.pdf*

7.4. Управление устройствами

7.4.1. Сетевое оборудование

Udev по умолчанию называет сетевые устройства в соответствии с данными прошивки, BIOS или физическими характеристиками, такими как шина, слот или MAC-адрес. Цель такого соглашения о именовании нужна для гарантии того, что сетевые устройства названы последовательно и не на основывались на времени обнаружении усстройства. Например, на компьютере с двумя сетевыми картами производства Intel и Realtek сетевая карта производства Intel может стать eth0, а карта Realtek-eth1. В некоторых случаях после перезагрузки, сетевые карты могут быть переименованы в другой последовательности.

В новой схеме именования обычные имена сетевых устройств будут чем-то вроде enp5s0 или wlp3s0. Если это соглашение о именовании не требуется, традиционная схема или своя собственная, может быть использована.

7.4.1.1. Отключение постоянного именования в командной строке ядра

Традиционная схема именования - eth0, eth1, и так далее, могут быть установлены путем добавления **`net.ifnames=0`** в командную строку ядра. Это подойдёт для систем, которые имеют только сетевое устройство одного типа. В ноутбуках часто есть несколько сетевых соединений с именами eth0 и wlan0. Эта схема вполне применима к ним. Командная строка передается в файле конфигурации GRUB. См. Section 8.4.4, “Создание файла конфигурации GRUB ”.

7.4.1.2. Создание собственных правил Udev

Именование схемы может быть настроено, созданием собственных правил Udev. Существует файл сценария, который генерирует начальные правила. Чтобы их сгенерировать, выполните команду:

```
bash /lib/udev/init-net-rules.sh
```

Теперь, проверьте файл `/etc/udev/rules.d/70-persistent-net.rules` , и найдите в нём данные о том, какое название с каким сетевым устройством сопоставлено:

```
cat /etc/udev/rules.d/70-persistent-net.rules
```

Note

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 создан, поскольку адреса не назначаются последовательно. В этих случаях, этот способ не применим.

Файл начинается с блока комментариев, далее следуют две строки для каждой сетевой карты (NIC). Первая строка, в блоке комментария - это её описание, аппаратные идентификаторы (например, поставщик PCI и идентификаторы устройств, если это PCI-карта), вместе со своим драйвером в скобках, если драйвер удаётся найти. Никакой идентификатор оборудования и драйвер не используются для определения имени интерфейса. Эта информация только для справки. Вторая строка - правило Udev, которое соответствует сетевой карте и фактически присваивает ему имя.

Все правила udev состоят из нескольких ключей, разделенных запятыми и необязательными пробелами. Ключи правила и объяснение каждого из них:

- SUBSYSTEM=="net" - сообщает Udev игнорировать устройства, которые не являются сетевыми картами.
- ACTION=="add" - сообщает Udev игнорировать это правило для события uevent которое не добавляет (события uevents "удалить" и "добавить" также произойдут, но без необходимости переименования сетевого интерфейса).
- DRIVERS=="?*" - Это сделано так, чтобы Udev проигнорировал подинтерфейсы VLAN или моста (потому что эти подинтерфейсы не имеют драйверов). Эти подинтерфейсы пропускаются, поскольку имя, которое было бы назначено, столкнулось бы с их родительскими устройствами.
- ATTR{address} - Значение этого ключа - является MAC адресом сетевой карты.
- ATTR{type}=="1" - Это гарантирует, что правило только совпадает с основным интерфейсом в случае определенных беспроводных драйверов, которые создают множественные виртуальные интерфейсы. Вторичные интерфейсы пропущены по той же причине, что VLAN и подинтерфейсы моста. В ином случае будет конфликт имен.
- NAME - Значение этого ключа - наименование сетевого интерфейса, назначенного Udev.

Значение NAME является важным. Прежде чем продолжить, убедитесь, что вы знаете, какое имя назначено каждой из сетевых карт, и не забудьте использовать это имя при создании файлов конфигурации ниже.

7.4.2. Символические ссылки CD-ROM

Некоторые программы, которые вы захотите установить позднее (например, различные медиа-проигрыватели), ожидают, устройства /dev/cdrom и /dev/dvd и символические ссылки на CD-ROM или DVD-ROM устройства. Кроме того, может быть удобно поместить ссылки на эти символические ссылки в /etc/fstab . Udev поставляется со файлом сценария, который будет генерировать файлы правил для создания этих символических ссылок для вас, 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 программами **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, которые он подключает), 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 умереть, и вы замените его другим устройством с теми же возможностями и которое подключено к тем же разъемам, тогда вы должны использовать режим “by-path”.

Если с вашим устройством возможен любой из вариантов, выберите тот, который случается наиболее часто.

Important

Внешние устройства (например, компакт-диск, подключенный через USB) не следует использовать методом “by-path”, потому что каждый раз, когда устройство подключено в новый внешний порт, изменится его физический путь. Все внешние устройства подвержены этой проблеме при написании правил Udev при распознавании их по их физическому пути. К тому же, эта проблема не ограничивается CD и DVD-дисками.

Если вы хотите увидеть значения, которые будут использоваться сценариями Udev, то для соответствующего устройства CD-ROM найдите в каталоге `/sys` (например, это может быть каталог `/sys/block/hdd`) и выполните команду, наподобие следующей:

```
udevadm test /sys/block/hdd
```

Обратите внимание на строки, содержащие вывод различных программ *_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 файле `/etc/udev/rules.d/83-cdrom-symlinks.rules` можно сделать изменение следующим образом (где *mode* один из “by-id” или “by-path”):

```
sed -i -e 's/"write_cd_rules"/"write_cd_rules mode"/' \
    /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` хост системы в системе LFS, и мы предполагаем, что символические ссылки уже существуют. Правила и символические ссылки будут созданы при первой загрузке LFS системы.

Однако, если у вас есть несколько устройств CD-ROM, то символические ссылки, сгенерированные в то 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 стабильными, поэтому проблема возникает 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.

7.4.3. Работа с дублирующими устройствами

Как поясняется в Section 7.3, “Обработка устройств и модулей”, 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, кроме звуковых и сетевых карт, это поправимо, путем создания пользовательских постоянных ссылкок udev. Случай относительно сетевых карт рассказаны отдельно в Section 7.5, “Конфигурация Сети”, и инструкции по конфигурированию 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 видео устройств, это могут быть каталоги `/sys/class/video4linux/videoX` . Выясните атрибуты, которые идентифицируют устройство однозначно (обычно, это идентификаторы поставщика и продукта и / или серийные номера):

```
udevadm info -a -p /sys/class/video4linux/video0
```

Затем, напишите правило, которое создаёт 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.

7.5. Конфигурация Сети

7.5.1. Создание файлов конфигурации сетевого интерфейса

Какие интерфейсы вызываются и выключаются сетевым сценарием, обычно зависит от файлов, которые находятся в каталоге `/etc/sysconfig/` . Этот каталог должен содержать файл для каждого настраиваемого интерфейса, например `ifconfig.xyz` , где "xyz" должен описывать сетевую карту. Имя интерфейса (например, eth0) обычно является подходящим. Внутри этого файла находятся атрибуты этого интерфейса, такие как IP-адреса, маски подсети и так далее. Необходимо, чтобы название файла начиналось с *ifconfig*.

Note

Если процедура в предыдущем разделе не использовалась, Udev назначит имена интерфейсов сетевых карт на основе физических характеристик системы, таких как enp2s1. Если Вы не уверены, каково ваше имя интерфейса, вы всегда можете запустить команду **ip link** или **ls /sys/class/net** после загрузки вашей системы.

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 этих переменных должны быть изменены в каждом файле, для правильной настройки и работы сети.

Если переменной ONBOOT указать значение “yes”, сценарий работы сети пакета System V вызовет сетевую карту в процессе загрузки системы. Если задано значение, отличное от “yes” , сетевой адаптер будет игнорироваться сценарием и не будет автоматически вызываться. Интерфейс можно запустить или остановить 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, и создание дополнительных файлов в каталоге /lib/services/ позволит использовать другие методы назначения IP. Как правило, используется для протокола DHCP (Dynamic Host Configuration Protocol), который рассматривается в книге BLFS.

Переменная GATEWAY должна содержать значение по умолчанию IP-адрес шлюза, если таковой имеется. Если нет, то закомментируйте строку.

Переменная PREFIX переменная содержит количество бит, используемых в подсети. Каждый байт в IP адресе - 8 бит. Если маска подсети 255.255.255.0, в таком случае, используется первые три байта (24 бита) для указания сетевого номера. Если маска подсети 255.255.255.240, то будет использовано первые 24 бита. Префиксы длиннее, чем 24 бита используются в основном провайдерами с cDSL и кабельными соединениями. В этом примере (PREFIX=24) маска подсети 255.255.255.0. Настройте переменную PREFIX в соответствии конфигурацией вашей подсети. Если опустить префикс, по умолчанию будет значение 24.

Для более подробной информации, прочитайте руководство **ifup**.

7.5.2. Создание файла /etc/resolv.conf

Системе потребуются некоторые средства для получения имен службы доменных имен (DNS) для преобразования доменных имен сети Интернет в IP-адреса и наоборот. Это достигается путем размещения 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 <Your Domain Name>
nameserver <IP address of your primary nameserver>
nameserver <IP address of your secondary nameserver>

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

Запись `domain` может быть опущена или заменена `search` записью. Прочитайте руководство resolv.conf для более подробной информации.

Замените *<IP address of the nameserver>* адресом наиболее подходящего DNS сервера. DNS серверов, может быть указано более одной записи (необходимо для вторичных серверов для возможности резервного переключения). Если вам нужен только один DNS-сервер, удалите вторую строку *nameserver* из файла. IP-адрес также может быть маршрутизатором в локальной сети.

Note

Адреса общедоступных DNS серверов Google - 8.8.8.8 и 8.8.4.4.

7.5.3. Настройка имени хоста

В процессе загрузки файл `/etc/hostname` используется для установки имени хоста системы.

Создайте файл `/etc/hostname` и внесите в имя хоста, выполнив команду:

```
echo "<lfs>" > /etc/hostname
```

<lfs> замените на имя вашего компьютера. Не вносите FQDN имя. Эта информация должна располагаться в файле `/etc/hosts` file.

7.5.4. Настройка файла /etc/hosts

Определите IP-адрес, полное доменное имя (FQDN) и возможные псевдонимы для использования в файле /etc/hosts. Синтаксис команды:

```
IP_address myhost.example.org aliases
```

Если компьютер не должен быть виден в Интернете (т. е. есть зарегистрированный домен и допустимый блок назначенных IP—адресов не имеются), убедитесь, что IP-адрес находится в диапазоне IP-адресов частной сети. Допустимые диапазоны:

```
Private Network Address Range      Normal Prefix
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 номером в пределах 16-31. y может быть номером в пределах 0-255.

Правильный IP адрес может быть 192.168.1.1. Правильный FQDN для этого IP адреса может быть lfs.example.org.

Даже если сетевая карта не используется, все равно требуется действительное полное доменное имя. Оно необходимо для правильной работы некоторых программ

Создайте файл `/etc/hosts` выполнив команду:

```
cat > /etc/hosts << "EOF"
# Begin /etc/hosts

127.0.0.1 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 и параметров сети (если имеется IP-адрес выданный системным / сетевым администратором и машина подключена к существующей сети.) Необязательные псевдонимы могут быть опущены.

7.6. Настройка и использование System V Bootscript

7.6.1. Как работает System V Bootscripts?

В Linux используется специальное средство загрузки SysVinit, основанное на концепции уровней выполнения (*run-levels*). Он может сильно отличаться от одной системы к другой, поэтому нельзя предположить, что, корректная работа в одном дистрибутиве Linux, должны работать одинаково и в LFS. У LFS есть свой собственный способ управления, но она уважает общепринятые стандарты.

SysVinit (который теперь будет называться “init”) работает по схеме уровней выполнения. Есть семь (пронумерованных от 0 до 6) уровней выполнения (на самом деле, есть больше уровней выполнения, но они предназначены для особых случаев и обычно не используются. См. `init(8)`

Для получения более подробной информации), и каждый из них соответствует действиям, которые компьютер должен выполнять при запуске. Уровень запуска по умолчанию-3. Ниже приведены описания различных уровней запуска по мере их реализации:

```
0: остановить компьютер
1: однопользовательский режим
2: многопользовательский режим без сети
3: многопользовательский режим с сетью
4: зарезервированный для настроек, в ином случае аналогичен уровню 3
5: аналогичен уровню 3 4, обычно используется для GUI авторизации (например xdm или kdm)
6: перезагрузка компьютера
```

7.6.2. Настройка Sysvinit

Во время инициализации ядра первая запущенная программа указывается либо в командной строке, либо по умолчанию в init. Эта программа читает 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:S016:once:/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
```

Объяснение содержимого файла инициализации находится на справочных страницах *inittab*. Для LFS ключевой командой является rc. В приведенном выше файле инициализации rc будет выполнять все сценарии, начиная с символа S в каталоге `/etc/rc.d/rcS.d`, за которым следуют все скрипты, начинающиеся с S в /etc/rc.d / rc?.D каталог, в котором знак вопроса задается значением initdefault.

Для удобства, сценарий **rc** считывает функции из библиотеки `/lib/lsb/init-functions`. Библиотека, в свою очередь, считывает опциональные файлы конфигурации из файла `/etc/sysconfig/rc.site`. Любой из системных конфигурационных файлов описанных в последующих разделах можно, в качестве альтернативы, поместить в этот файл, что позволит объединить все системные параметры в один файл.

Для удобства отладки, функции сценария журналируют весь вывод в файл `/run/var/bootlog`. Поскольку каталог `/run` является tmpfs, файла может не быть, в процессе загрузки, однако уго содержимое добавляется к фиксированному файлу `/var/log/boot.log` при завершении процесса загрузки.

7.6.2.1. Изменение уровней выполнения (Run Levels)

Изменить уровень выполнения можно с помощью команды **init *<runlevel>***, где *<runlevel>* - будет целевым уровнем выполнения. Например, чтобы перезагрузить компьютер, пользователь должен выполнить команду **init 6**, которая является псевдонимом команды **reboot**. Аналогично, init 0 является псевдонимом команды **halt**.

В каталоге `/etc/rc.d` есть ещё несколько каталогов, которые выглядят как `rc?.d` (где ? номер уровня выполнения) и каталог `rcsysinit.d`, содержащий некоторый набор символических ссылок. Некоторые начинаются с K, другие начинаются с S, и все они имеют два числа после начальной буквы. K означает остановить (kill) службу, а S означает запустить службу. Числа определяют порядок выполнения сценариев от 00 до 99 - чем меньше число, тем раньше он запустится. Когда **init** переключается на другой уровень выполнения, соответствующие службы запускаются или останавливаются в зависимости от выбранного уровня запуска.

Реальные сценарии находятся в каталоге `/etc/rc.d/init.d`. Они выполняют фактическую работу, и символические ссылки указывают на них. Ссылки K и ссылки S указывают на тот же сценарий в `/etc/rc.d/init.d`. Это связано с тем, что сценарии можно вызывать с различными 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`*.

Есть одно исключение. Ссылки, которые начинаются с *S* в каталогах `rc0.d` и `rc6.d`, не будут запускать что-либо. Они будут вызываться с параметром *`stop`*, для того, чтобы остановить какую либо службу. Логика этого заключается в том, что когда пользователь собирается перезагрузить или остановить систему, при этом, ему ничего не нужно запускать. Систему нужно только остановить.

Ниже приведены описания к аргументам для сценариев и их значения:

`start`
: Служба запущена

`stop`
: Служба остановлена

restart
Остановка службы и затем повторный её запуск

reload
Конфигурация службы была обновлена. Используется после того, как файлы конфигурации были обновлены, когда службу не нужно перезапускать.

status
Сообщает, работает ли служба и какие ей назначены PID

Не стесняйтесь изменять способ загрузки процесса (в конце концов, это ваша собственная система LFS). Приведенные здесь файлы являются примером того, как это можно сделать.

7.6.3. Udev Bootscripts

The `/etc/rc.d/init.d/udev` initscript starts **udevd**, triggers any "coldplug" devices that have already been created by the kernel and waits for any rules to complete. The script also unsets the uevent handler from the default of `/sbin/hotplug`. This is done because the kernel no longer needs to call out to an external binary. Instead **udevd** will listen on a netlink socket for uevents that the kernel raises.

The **/etc/rc.d/init.d/udev_retry** initscript takes care of re-triggering events for subsystems whose rules may rely on filesystems that are not mounted until the **mountfs** script is run (in particular, `/usr` and `/var` may cause this). This script runs after the **mountfs** script, so those rules (if re-triggered) should succeed the second time around. It is configured from the `/etc/sysconfig/udev_retry` file; any words in this file other than comments are considered subsystem names to trigger at retry time. To find the subsystem of a device, use **udevadm info --attribute-walk <device>** where <device> is an absolute path in /dev or /sys such as /dev/sr0 or /sys/class/rtc.

For information on kernel module loading and udev, see Section 7.3.2.3, “Загрузка модулей”.

7.6.4. Configuring the System Clock

The **setclock** script reads the time from the hardware clock, also known as the BIOS or the Complementary Metal Oxide Semiconductor (CMOS) clock. If the hardware clock is set to UTC, this script will convert the hardware clock's time to the local time using the `/etc/localtime` file (which tells the **hwclock** program which timezone the user is in). There is no way to detect whether or not the hardware clock is set to UTC, so this needs to be configured manually.

The **setclock** is run via udev when the kernel detects the hardware capability upon boot. It can also be run manually with the stop parameter to store the system time to the CMOS clock.

If you cannot remember whether or not the hardware clock is set to UTC, find out by running the **hwclock --localtime --show** command. This will display what the current time is according to the hardware clock. If this time matches whatever your watch says, then the hardware clock is set to local time. If the output from **hwclock** is not local time, chances are it is set to UTC time. Verify this by adding or subtracting the proper amount of hours for the timezone to the time shown by **hwclock**. For example, if you are currently in the MST timezone, which is also known as GMT -0700, add seven hours to the local time.

Change the value of the UTC variable below to a value of *0* (zero) if the hardware clock is *not* set to UTC time.

Create a new file `/etc/sysconfig/clock` by running the following:

```
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
```

A good hint explaining how to deal with time on LFS is available at *https://linuxfromscratch.ru/hints/downloads/files/time.txt*. It explains issues such as time zones, UTC, and the TZ environment variable.

Note

The CLOCKPARAMS and UTC paramaters may be alternatively set in the `/etc/sysconfig/rc.site` file.

7.6.5. Configuring the Linux Console

This section discusses how to configure the **console** bootscript that sets up the keyboard map, console font and console kernel log level. If non-ASCII characters (e.g., the copyright sign, the British pound sign and Euro symbol) will not be used and the keyboard is a U.S. one, much of this section can be skipped. Without the configuration file, (or equivalent settings in `rc.site`), the **console** bootscript will do nothing.

The **console** script reads the `/etc/sysconfig/console` file for configuration information. Decide which keymap and screen font will be used. Various language-specific HOWTOs can also help with this, see *http://www.tldp.org/HOWTO/HOWTO-INDEX/other-lang.html*. If still in doubt, look in the `/usr/share/keymaps` and `/usr/share/consolefonts` directories for valid keymaps and screen fonts. Read `loadkeys(1)` and `setfont(8)` manual pages to determine the correct arguments for these programs.

The `/etc/sysconfig/console` file should contain lines of the form: VARIABLE="value". The following variables are recognized:

LOGLEVEL
: This variable specifies the log level for kernel messages sent to the console as set by **dmesg**. Valid levels are from "1" (no messages) to "8". The default level is "7".

KEYMAP
: This variable specifies the arguments for the **loadkeys** program, typically, the name of keymap to load, e.g., “it”. If this variable is not set, the bootscript will not run the **loadkeys** program, and the default kernel keymap will be used. Note that a few keymaps have multiple versions with the same name (cz and its variants in qwerty/ and qwertz/, es in olpc/ and qwerty/, and trf in fgGIod/ and qwerty/). In these cases the parent directory should also be specified (e.g. qwerty/es) to ensure the proper keymap is loaded.

KEYMAP_CORRECTIONS
: This (rarely used) variable specifies the arguments for the second call to the **loadkeys** program. This is useful if the stock keymap is not completely satisfactory and a small adjustment has to be made. E.g., to include the Euro sign into a keymap that normally doesn't have it, set this variable to “euro2”.

FONT
: This variable specifies the arguments for the **setfont** program. Typically, this includes the font name, “-m”, and the name of the application character map to load. E.g., in order to load the “lat1-16” font together with the “8859-1” application character map (as it is appropriate in the USA), set this variable to “lat1-16 -m 8859-1”. In UTF-8 mode, the kernel uses the application character map for conversion of composed 8-bit key codes in the keymap to UTF-8, and thus the argument of the "-m" parameter should be set to the encoding of the composed key codes in the keymap.

UNICODE
: Set this variable to “1”, “yes” or “true” in order to put the console into UTF-8 mode. This is useful in UTF-8 based locales and harmful otherwise.

LEGACY_CHARSET
: For many keyboard layouts, there is no stock Unicode keymap in the Kbd package. The **console** bootscript will convert an available keymap to UTF-8 on the fly if this variable is set to the encoding of the available non-UTF-8 keymap.

Some examples:

- For a non-Unicode setup, only the KEYMAP and FONT variables are generally needed. E.g., for a Polish setup, one would use:

```
cat > /etc/sysconfig/console << "EOF"
# Begin /etc/sysconfig/console

KEYMAP="pl2"
FONT="lat2a-16 -m 8859-2"

# End /etc/sysconfig/console
EOF
```

- As mentioned above, it is sometimes necessary to adjust a stock keymap slightly. The following example adds the Euro symbol to the German keymap:

```
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
```

- The following is a Unicode-enabled example for Bulgarian, where a stock UTF-8 keymap exists:

```
cat > /etc/sysconfig/console << "EOF"
# Begin /etc/sysconfig/console

UNICODE="1"
KEYMAP="bg_bds-utf8"
FONT="LatArCyrHeb-16"

# End /etc/sysconfig/console
EOF
```

- Due to the use of a 512-glyph LatArCyrHeb-16 font in the previous example, bright colors are no longer available on the Linux console unless a framebuffer is used. If one wants to have bright colors without framebuffer and can live without characters not belonging to his language, it is still possible to use a language-specific 256-glyph font, as illustrated below:

```
cat > /etc/sysconfig/console << "EOF"
# Begin /etc/sysconfig/console

UNICODE="1"
KEYMAP="bg_bds-utf8"
FONT="cyr-sun16"

# End /etc/sysconfig/console
EOF
```

- The following example illustrates keymap autoconversion from ISO-8859-15 to UTF-8 and enabling dead keys in Unicode mode:

```
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
```

- Some keymaps have dead keys (i.e., keys that don't produce a character by themselves, but put an accent on the character produced by the next key) or define composition rules (such as: “press Ctrl +. A E to get Æ” in the default keymap). Linux-5.2.8 interprets dead keys and composition rules in the keymap correctly only when the source characters to be composed together are not multibyte. This deficiency doesn't affect keymaps for European languages, because there accents are added to unaccented ASCII characters, or two ASCII characters are composed together. However, in UTF-8

mode it is a problem, e.g., for the Greek language, where one sometimes needs to put an accent on the letter “alpha”. The solution is either to avoid the use of UTF-8, or to install the X window system that doesn't have this limitation in its input handling.

- For Chinese, Japanese, Korean and some other languages, the Linux console cannot be configured to display the needed characters. Users who need such languages should install the X Window System, fonts that cover the necessary character ranges, and the proper input method (e.g., SCIM, it supports a wide variety of languages).

Note

The `/etc/sysconfig/console` file only controls the Linux text console localization. It has nothing to do with setting the proper keyboard layout and terminal fonts in the X Window System, with ssh sessions or with a serial console. In such situations, limitations mentioned in the last two list items above do not apply.

7.6.6. Creating Files at Boot

At times, it is desired to create files at boot time. For instance, the `/tmp/.ICE-unix` directory may be desired. This can be done by creating an entry in the `/etc/sysconfig/createfiles` configuration script. The format of this file is embedded in the comments of the default configuration file.

7.6.7. Configuring the sysklogd Script

The `sysklogd` script invokes the **syslogd** program as a part of System V initialization. The *`-m 0`* option turns off the periodic timestamp mark that **syslogd** writes to the log files every 20 minutes by default. If you want to turn on this periodic timestamp mark, edit `/etc/sysconfig/rc.site` and define the variable SYSKLOGD_PARMS to the desired value. For instance, to remove all parameters, set the variable to a null value:

```
SYSKLOGD_PARMS=
```

See **`man syslogd`** for more options.

7.6.8. Файл rc.site

The optional `/etc/sysconfig/rc.site` file contains settings that are automatically set for each SystemV boot script. It can alternatively set the values specified in the `hostname`, `console`, and `clock` files in the `/etc/sysconfig/` directory. If the associated variables are present in both these separate files and `rc.site` , the values in the script specific files have precedence.

`rc.site` also contains parameters that can customize other aspects of the boot process. Setting the IPROMPT variable will enable selective running of bootscripts. Other options are described in the file comments. The default version of the file is as follows:

```
# rc.site
# Optional parameters for boot scripts.

# Distro Information
# These values, if specified here, override the defaults
#DISTRO="Linux From Scratch" # The distro name
```

```
#DISTRO_CONTACT="lfs-dev@linuxfromscratch.org" # Bug report address
#DISTRO_MINI="LFS" # Short name used in filenames for distro config

# Define custom colors used in messages printed to the screen

# Please consult `man console_codes` for more information
# under the "ECMA-48 Set Graphics Rendition" section
#
# Warning: when switching from a 8bit to a 9bit font,
# the linux console will reinterpret the bold (1;) to
# the top 256 glyphs of the 9bit font.  This does
# not affect framebuffer consoles

# These values, if specified here, override the defaults
#BRACKET="\\033[1;34m" # Blue
#FAILURE="\\033[1;31m" # Red
#INFO="\\033[1;36m"    # Cyan
#NORMAL="\\033[0;39m"  # Grey
#SUCCESS="\\033[1;32m" # Green
#WARNING="\\033[1;33m" # Yellow

# Use a colored prefix
# These values, if specified here, override the defaults
#BMPREFIX="      "
#SUCCESS_PREFIX="${SUCCESS}  *  ${NORMAL} "
#FAILURE_PREFIX="${FAILURE}*****${NORMAL} "
#WARNING_PREFIX="${WARNING} *** ${NORMAL} "

# Manually seet 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=
```

7.6.8.1. Customizing the Boot and Shutdown Scripts

The LFS boot scripts boot and shut down a system in a fairly efficient manner, but there are a few tweaks that you can make in the rc.site file to improve speed even more and to adjust messages according to your preferences. To do this, adjust the settings in the `/etc/sysconfig/rc.site` file above.

- During the boot script udev, there is a call to **udev settle** that requires some time to complete. This time may or may not be required depending on devices present in the system. If you only have simple partitions and a single ethernet card, the boot process will probably not need to wait for this command. To skip it, set the variable OMIT_UDEV_SETTLE=y.
- The boot script `udev_retry` also runs **udev settle** by default. This command is only needed by default if the `/var` directory is separately mounted. This is because the clock needs the file `/var/lib/hwclock/adjtime`. Other customizations may also need to wait for udev to complete, but in many installations it is not needed. Skip the command by setting the variable OMIT_UDEV_RETRY_SETTLE=y.
- By default, the file system checks are silent. This can appear to be a delay during the bootup process. To turn on the **fsck** output, set the variable VERBOSE_FSCK=y.
- When rebooting, you may want to skip the filesystem check, **fsck**, completely. To do this, either create the file `/fastboot` or reboot the system with the command **/sbin/shutdown -f -r now**. On the other hand, you can force all file systems to be checked by creating `/forcefsck` or running **shutdown** with the *-F* parameter instead of *-f*.

  Setting the variable FASTBOOT=y will disable **fsck** during the boot process until it is removed. This is not recommended on a permanent basis.
- Normally, all files in the `/tmp` directory are deleted at boot time. Depending on the number of files or directories present, this can cause a noticeable delay in the boot process. To skip removing these files set the variable SKIPTMPCLEAN=y.
- During shutdown, the **init** program sends a TERM signal to each program it has started (e.g. agetty), waits for a set time (default 3 seconds), and sends each process a KILL signal and waits again. This process is repeated in the **sendsignals** script for any processes that are not shut down by their own scripts. The delay for **init** can be set by passing a parameter. For example to remove the delay in **init**, pass the -t0 parameter when shutting down or rebooting (e.g. **/sbin/shutdown -t0 -r now**). The delay for the **sendsignals** script can be skipped by setting the parameter KILLDELAY=0.

7.7. Файлы запуска оболочки Bash

The shell program **/bin/bash** (hereafter referred to as “the shell”) uses a collection of startup files to help create an environment to run in. Each file has a specific use and may affect login and interactive environments differently. The files in the `/etc` directory provide global settings. If an equivalent file exists in the home directory, it may override the global settings.

An interactive login shell is started after a successful login, using **/bin/login**, by reading the `/etc/passwd` file. An interactive non-login shell is started at the command-line (e.g., `[prompt]$` **/bin/bash**). A non-interactive shell is usually present when a shell script is running. It is non-interactive because it is processing a script and not waiting for user input between commands.

For more information, see **info bash** under the *Bash Startup Files and Interactive Shells* section.

The files `/etc/profile` and `~/.bash_profile` are read when the shell is invoked as an interactive login shell.

The base `/etc/profile` below sets some environment variables necessary for native language support. Setting them properly results in:

- The output of programs translated into the native language
- Correct classification of characters into letters, digits and other classes. This is necessary for **bash** to properly accept non-ASCII characters in command lines in non-English locales
- The correct alphabetical sorting order for the country
- Appropriate default paper size
- Correct formatting of monetary, time, and date values

Replace *<ll>* below with the two-letter code for the desired language (e.g., "en") and *<CC>* with the two-letter code for the appropriate country (e.g., "GB"). *<charmap>* should be replaced with the canonical charmap for your chosen locale. Optional modifiers such as "@euro" may also be present.

The list of all locales supported by Glibc can be obtained by running the следующую команду:

```
locale -a
```

Charmaps can have a number of aliases, e.g., "ISO-8859-1" is also referred to as "iso8859-1" and "iso88591". Some applications cannot handle the various synonyms correctly (e.g., require that "UTF-8" is written as "UTF-8", not "utf8"), so it is safest in most cases to choose the canonical name for a particular locale. To determine the canonical name, run the following command, where *<locale name>* is the output given by **locale -a** for your preferred locale ("en_GB.iso88591" in our example).

```
LC_ALL=<locale name> locale charmap
```

For the "en_GB.iso88591" locale, the above command will print:

```
ISO-8859-1
```

This results in a final locale setting of "en_GB.ISO-8859-1". It is important that the locale found using the heuristic above is tested prior to it being added to the Bash startup files:

```
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
```

The above commands should print the language name, the character encoding used by the locale, the local currency, and the prefix to dial before the telephone number in order to get into the country. If any of the commands above fail with a message similar to the one shown below, this means that your locale was either not installed in Chapter 6 or is not supported by the default installation of Glibc.

```
locale: Cannot set LC_* to default locale: No such file or directory
```

If this happens, you should either install the desired locale using the **localedef** command, or consider choosing a different locale. Further instructions assume that there are no such error messages from Glibc.

Some packages beyond LFS may also lack support for your chosen locale. One example is the X library (part of the X Window System), which outputs the following error message if the locale does not exactly match one of the character map names in its internal files:

```
Warning: locale not supported by Xlib, locale set to C
```

In several cases Xlib expects that the character map will be listed in uppercase notation with canonical dashes. For instance, "ISO-8859-1" rather than "iso88591". It is also possible to find an appropriate specification by removing the charmap part of the locale specification. This can be checked by running the **locale charmap** command in both locales. For example, one would have to change "de_DE.ISO-8859-15@euro" to "de_DE@euro" in order to get this locale recognized by Xlib.

Other packages can also function incorrectly (but may not necessarily display any error messages) if the locale name does not meet their expectations. In those cases, investigating how other Linux distributions support your locale might provide some useful information.

Once the proper locale settings have been determined, create the `/etc/profile` file:

```
cat > /etc/profile << "EOF"
# Begin /etc/profile

export LANG=<ll>_<CC>.<charmap><@modifiers>

# End /etc/profile
EOF
```

The “C” (default) and “en_US” (the recommended one for United States English users) locales are different. “C” uses the US-ASCII 7-bit character set, and treats bytes with the high bit set as invalid characters. That's why, e.g., the **ls** command substitutes them with question marks in that locale. Also, an attempt to send mail with such characters from Mutt or Pine results in non-RFC-conforming messages being sent (the charset in the outgoing mail is indicated as “unknown 8-bit”). So you can use the “C” locale only if you are sure that you will never need 8-bit characters.

UTF-8 based locales are not supported well by some programs. Work is in progress to document and, if possible, fix such problems, see *https://linuxfromscratch.ru/blfs/view/svn/introduction/locale-issues.html*.

7.8. Создание файла /etc/inputrc File

Файл inputrc-это файл конфигурации библиотеки Readline, который предоставляет возможности редактирования при вводе пользователем строки из терминала. Он перенаправляет ввода с клавиатуры в конкретные действия. Readline используется в Bash и большинством других оболочек, а также многими другими приложениями.

Большинство людей не нуждаются в настраиваемой вручную функциональности, поэтому команда ниже создает глобальный `/etc/inputrc` , используемый всеми, кто входит в систему. Если вы позднее решите, что необходимо переопределить значения по умолчанию, можно создать файл `.inputrc` в домашнем каталоге пользователя с измененными значениями.

Дополнительные сведения о редактировании файла `inputrc` см. в разделе **info bash** в секции *Readline Init File*.**info readline** также является хорошим источником информации.

Ниже приведен общий глобальный inputrc вместе с комментариями, чтобы объяснить, что делают различные варианты. Обратите внимание, что комментарии не могут находиться в одной строке с командами. Создайте файл, выполнив следующую команду:

```
cat > /etc/inputrc << "EOF"
# Begin /etc/inputrc
# Modified by Chris Lynn <roryo@roryo.dynup.net>

# Allow the command prompt to wrap to the next line
set horizontal-scroll-mode Off

# Enable 8bit input
set meta-flag On
set input-meta On

# Turns off 8th bit stripping
set convert-meta Off

# Keep the 8th bit for display
set output-meta On

# none, visible or audible
set bell-style none

# All of the following map the escape sequence of the value
# contained in the 1st argument to the readline specific functions
"\eOd": backward-word
"\eOc": forward-word

# for linux console
"\e[1~": beginning-of-line
"\e[4~": end-of-line
"\e[5~": beginning-of-history
"\e[6~": end-of-history
"\e[3~": delete-char
"\e[2~": quoted-insert

# for xterm
"\eOH": beginning-of-line
"\eOF": end-of-line

# for Konsole
"\e[H": beginning-of-line
"\e[F": end-of-line

# End /etc/inputrc
EOF
```

7.9. 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, пользователю будет отказано в изменении.

Это требование для таких приложений, как GDM, которые не заполняют браузер face, если он не удается найти /etc/shells , или FTP-демоны, 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
```

Chapter 8. Делаем систему LFS загрузочной

8.1. Введение

Самое время сделать систему LFS загрузочной. В этой главе обсуждается создание файла `fstab`, сборка ядра для новой системы LFS и установка загрузчика GRUB таким образом, чтобы при начальной загрузке, новую LFS систему можно было выбрать.

8.2. Создание файла /etc/fstab

The `/etc/fstab` file is used by some programs to determine where file systems are to be mounted by default, in which order, and which must be checked (for integrity errors) prior to mounting. Create a new file systems table like this:

```
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

# End /etc/fstab
EOF
```

Replace *<xxx>*, *<yyy>*, and *<fff>* with the values appropriate for the system, for example, `sda2`, `sda5`, and `ext4`. For details on the six fields in this file, see **man 5 fstab**.

Filesystems with MS-DOS or Windows origin (i.e. vfat, ntfs, smbfs, cifs, iso9660, udf) need a special option, utf8, in order for non-ASCII characters in file names to be interpreted properly. For non-UTF-8 locales, the value of `iocharset` should be set to be the same as the character set of the locale, adjusted in such a way that the kernel understands it. This works if the relevant character set definition (found under File systems -> Native Language Support when configuring the kernel) has been compiled into the kernel or built as a module. However, if the character set of the locale is UTF-8, the corresponding option `iocharset=utf8` would make the file system case sensitive. To fix this, use the special option `utf8` instead of `iocharset=utf8`, for UTF-8 locales. The “codepage” option is also needed for vfat and smbfs filesystems. It should be set to the codepage number used under MS-DOS in your country. For example, in order to mount USB flash drives, a ru_RU.KOI8-R user would need the following in the options portion of its mount line in `/etc/fstab` :

```
noauto,user,quiet,showexec,codepage=866,iocharset=koi8r
```

The corresponding options fragment for ru_RU.UTF-8 users is:

```
noauto,user,quiet,showexec,codepage=866,utf8
```

Note that using `iocharset` is the default for `iso8859-1` (which keeps the file system case insensitive), and the `utf8` option tells the kernel to convert the file names using UTF-8 so they can be interpreted in the UTF-8 locale.

It is also possible to specify default codepage and iocharset values for some filesystems during kernel configuration. The relevant parameters are named “Default NLS Option” (`CONFIG_NLS_DEFAULT`), “Default Remote NLS Option” (`CONFIG_SMB_NLS_DEFAULT`), “Default codepage for FAT” (`CONFIG_FAT_DEFAULT_CODEPAGE`), and “Default iocharset for FAT” (`CONFIG_FAT_DEFAULT_IOCHARSET`). There is no way to specify these settings for the ntfs filesystem at kernel compilation time.

It is possible to make the ext3 filesystem reliable across power failures for some hard disk types. To do this, add the `barrier=1` mount option to the appropriate entry in `/etc/fstab`. To check if the disk drive supports this option, run *hdparm* on the applicable disk drive. For example, if:

```
hdparm -I /dev/sda | grep NCQ
```

returns non-empty output, the option is supported.

Note: Logical Volume Management (LVM) based partitions cannot use the `barrier` option.

8.3. Linux-5.2.8

Ядро операционной системы.

Приблизительное время сборки: 4.4 - 66.0 SBU (typically about 6 SBU)
Требуемое дисковое пространство: 960 - 4250 MB (typically about 1100 MB)

8.3.1. Установка ядра

Процесс сборки ядра состоит из нескольких этапов: настройка, компиляция и установка. Ознакомьтесь с файлом README в дереве исходных кодов пакета чтобы узнать о других способах настройки ядра.

Подготовьте пакет к компиляции выполнив следующую команду:

```
make mrproper
```

Выполнение этой команды гарантирует, что дерево исходных кодов ядра абсолютно чистое. Разработчики ядра рекомендуют, чтобы эта команда выполнялась перед каждым процессом компиляции. Обратите внимание что после распаковки пакета с исходным кодом не следует полагаться на его "чистоту".

Настройте ядро с помощью псевдографического интерфейса. Для получения общей информации перейдите по ссылке *https://linuxfromscratch.ru/hints/downloads/files/kernel-configuration.txt*. В книге BLFS есть информация по поводу некоторых требований к конфигурации ядра для поддержки пакетов, которые отсутствуют в книге LFS. Эта информация доступна по ссылке *https://linuxfromscratch.ru/blfs/view/svn/longindex.html#kernel-config-index*. Дополнительная информация о настройке и сборке ядра расположена по ссылке *http://www.kroah.com/lkn/*

Note

Хорошей отправной точкой для настройки ядра, может стать запуск команды **make defconfig**. В результате её выполнения будет создана базовая конфигурация с учётом архитектуры машины.

Убедитесь в том, что вы включили/отключили/указали указанные ниже параметры настройки, в ином случае, система может работать не правильно, или вовсе не загрузится:

```
Device Drivers  --->
  Generic Driver Options  --->
   [ ] Support for uevent helper [CONFIG_UEVENT_HELPER]
   [*] Maintain a devtmpfs filesystem to mount at /dev [CONFIG_DEVTMPFS]

Kernel hacking  --->
       Choose kernel unwinder (Frame pointer unwinder)  ---> [CONFIG_UNWINDE
```

Также, есть некоторые опции, которые могут понадобиться в зависимости от конфигурации системы. Список параметров, необходимых для пакетов в книге BLFS - *BLFS*.

Note

Если на хост системе используется UEFI, то после выполнения команды 'make defconfig' должны автоматически примениться параметры ядра, связанные с EFI.

Для того, чтобы ядро LFS могло загружаться из среды UEFI, необходимо включить соответствующий параметр:

```
Processor type and features  --->
   [*]   EFI stub support  [CONFIG_EFI_STUB]
```

Более полное описание управления UEFI из LFS доступно по ссылке: *http://www.linuxfromscratch.org/hints/downloads/files/lfs-uefi.txt*.

Разъяснение указанных параметров конфигурации:

Support for uevent helper
Наличие этого параметра может вызвать конфликт при управление устройствами через Udev / Eudev.

Maintain a devtmpfs
При использовании этого параметра, будет осуществлена поддержка узлов устройств, заполняемых самим ядром, даже без запуска Udev. Udev будет работать поверх, управляя разрешениями и добавляя необходимые символические ссылки. Этот элемент конфигурации необходим всем пользователям Udev / Eudev.

```
make menuconfig
```

Значение необязательных переменных окружения программы make:

LANG=<host_LANG_value> LC_ALL=
Задаёт настройку локали на ту, которая используется в хост системе для корректного отображения menuconfig при использовании интерфейса ncurses при генерации строк на текстовой консоли Linux с кодировкой UTF-8.
Если используете, то убедитесь что заменили переменную *<host_LANG_value>* на значение переменной окружения `$LANG` их хост системы. Или можно использовать значение переменной окружения хос системы `$LC_ALL` или `$LC_CTYPE` .

Кроме того, команда **make oldconfig** может быть более уместна в некоторых случаях. Дополнительную информацию см. В файле `README`.

При желании, можно пропустить этап настройки, скопировав файл конфигурации ядра `.config` с вашей хост системы (в том случае, если такой файл присутствует) в предварительно распакованный каталог ядра `linux-5.2.8` . Мы не рекомендуем такой подход. Гораздо полезнее изучить все параметры настройки и создать файл конфигурации ядра самостоятельно.

Скомпилируйте образ ядра и модули:

```
make
```

При использовании модулей, могут потребоваться файлы конфигурации, которые должны находиться в каталоге `/etc/modprobe.d` . Информация о модулях и конфигурации ядра находится в Section 7.3, “Обработка устройств и модулей” и в каталоге с документацией ядра `linux-5.2.8/Documentation` . Также будет интересным изучение информации `modprobe.d(5)` .

Установите модули, если ядро их использует:

```
make modules_install
```

После завершения компиляции, необходимо выполнить еще несколько шагов. Некоторые файлы должны быть скопированы в каталог `/boot` .

Caution

Если хост система содержит отдельный раздел для каталога /boot, файлы скопированные ниже, должны находится в нём. Самый простой способ это сделать - выполнить привязку (bind) каталога /boot хост системы (за пределами выполнения в среде chroot) к каталогу /mnt/lfs/boot перед тем, как продолжить. Из-под пользователя root в *хост системе* выполните команду:

```
mount --bind /boot /mnt/lfs/boot
```

Путь к образу ядра зависит от используемой платформы. Имя файла, указанное ниже, может иметь произвольное наименование, на ваш вкус, но имя файла должно начинаться с *vmlinuz* для обеспечения совместимости автоматической настройки процесса загрузки, описанного в следующей главе. При выполнении следующей команды, будет считать что используется архитектура x86:

```
cp -iv arch/x86/boot/bzImage /boot/vmlinuz-5.2.8-lfs-9.0
```

`System.map` файл, внутри которого находится символьная таблица адресов функций и процедур, используемых ядром операционной системы Linux. В этой таблице перечислены имена переменных и функций и их адреса в памяти компьютера. Эта таблица весьма полезна при отладке ядра в случае Kernel panic или Linux oops. System.map генерируется при компиляции ядра. Выполните следующщую команду для установки файла `System.map` :

```
cp -iv System.map /boot/System.map-5.2.8
```

Файл конфигурации ядра `.config` полученный в результате настройки **make menuconfig** содержит в себе все опции конфигурации скомпилированного ядра. Хорошей идеей будет оставить этот файл для будующей работы:

```
cp -iv .config /boot/config-5.2.8
```

Установите документацию ядра:

```
install -d /usr/share/doc/linux-5.2.8
cp -r Documentation/* /usr/share/doc/linux-5.2.8
```

Важно отметить, что файлы в каталоге исходных кодов ядра не принадлежат пользователю *root*. Всякий раз, когда пакет распаковывается от пользователя *root* (как это и выполнялось внутри среды chroot), файлы имеют те идентификаторы пользователя и группы, которые были назначены при распаковке. Обычно это не вызывает проблем для других устанавливаемых пакетов, так как каталог с исходными кодами 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.

Note

Во многих случаях конфигурация ядра должна быть обновлена для пакетов, которые будут установлены позже в BLFS. В отличие от других пакетов удалять дерево исходного кода ядра не требуется после компиляции и установки.

Если вы планируете оставить каталог с исходным кодом ядра, выполните команду: **chown -R 0:0** для каталога `linux-5.2.8` чтобы указать права для пользователя *root*.

Warning

В документации ядра рекомендуется создать символическую ссылку `/usr/src/linux` для указания местоположения каталога с исходными кодами ядра. Это 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 создать, когда ваша базовая система LFS будет готова.

Warning

Заголовочные файлы в системном каталоге `include` (`/usr/include`) должны *всегда* быть те, которые использовались при компиляции Glibc в главе Section 6.7, “Заголовочные файлы API Linux-5.2.8”. Поэтому их *никогда* не следует заменять на чистые заголовочные файлы ядра или любые другие подготовленные заголовочные файлы.

8.3.2. Настройка порядка загрузки модулей Linux

Обычно 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 создан так, что если USB-драйверы (ehci_hcd, ohci_hcd и uhci_hcd) были созданы в виде модулей, то они будут загружены в требуемом 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
```

8.3.3. Содержимое пакета Linux

Установленные файлы: config-5.2.8, vmlinuz-5.2.8-lfs-9.0, and System.map-5.2.8

Установленные каталоги: /lib/modules, /usr/share/doc/linux-5.2.8

Краткое описание

`config-5.2.8` Файл конфигурации ядра `.config` содержит в себе все опции конфигурации скомпилированного ядра.

`vmlinuz-5.2.8-lfs-9.0` Ядро системы Linux. При включении компьютера ядро - первая часть операционной системы, которая будет загружена. Она обнаруживает и инициализирует все компоненты оборудования компьютера, делает их доступными в виде дерева каталогов с файлами для доступа к ним программам и превращает один процессор в мультизадачную машину, способную выполнять множество программ как будто одновременно.

`System.map-5.2.8` файл, внутри которого находится символьная таблица адресов функций и процедур, используемых ядром операционной системы Linux. В этой таблице перечислены имена переменных и функций и их адреса в памяти компьютера. Эта таблица весьма полезна при отладке ядра в случае Kernel panic или Linux oops. System.map генерируется при компиляции ядра.

8.4. Использование GRUB для настройки процесса загрузки

8.4.1. Введение

Warning

Если вы настроите GRUB не правильно, то возможно, не получится штатно загрузиться в систему, без использования другого загрузочного устройства (например компакт-диска). Для загрузки системы LFS, эта глава не обязательна. Вы можете просто модифицировать ваш текущий системный загрузчик, например Grub-Legacy, GRUB2, или LILO.

Убедитесь, что аварийный загрузочный диск готов «спасти» ваш компьютер, если он перестанет правильно загружаться. Если у вас еще нет загрузочного устройства, вы можете его создать. Чтобы нижеследующая процедура работала, вам нужно перейти в 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
```

Note

Чтобы загрузить LFS в хост-системах с UEFI, ядро должно быть собрано с указанием соответствующего ключа конфигурации CONFIG_EFI_STUB. О сборке ядра, наиболее подробно, обсуждается в предыдущем разделе. Однако LFS можно загружать с помощью GRUB2 без такого дополнения. Для этого необходимо отключить функции UEFI Mode и Secure Boot в BIOS хост-системе. Для получения дополнительной информации см. *the lfs-uefi.txt подсказку*.

8.4.2. Соглашения о именовании в GRUB

GRUB использует собственную структуру именования дисков и разделов в виде *(hdn,m)*, где *n* - номер диска, а *m* - номер раздела. Номер диска начинается с нуля, а номер раздела начинается с единицы, для обычных разделов и с пяти для расширенных разделов. Обратите внимание, что именование отличается от предыдущих версий, когда оба номера начинались с нуля. Например, раздел sda1 равен *(hd0,1)* в GRUB, а sdb3 - *(hd1,3)*. В отличие от Linux, GRUB не считает диски CD-ROM жесткими дисками. Например, если вы используете CD на hdb и второй жесткий диск на hdc, этот второй жесткий диск будет *(hd1)*.

8.4.3. Настройка

GRUB записывает данные на первый физический сектор жесткого диска. Эта область не является частью какой-либо файловой системы. Программы доступны как модули GRUB в загрузочном разделе в каталоге по умолчанию -/boot/grub/.

Расположение загрузочного раздела - это ваш выбор, его можно настроить. Одна рекомендация состоит в том, чтобы иметь отдельный небольшой (рекомендуемый размер - 100 МБ) раздел только для загрузочной информации. Таким образом, каждая сборка, будь то LFS или какой-

либо другой дистрибутив, может обращаться к тем же загрузочным файлам, и доступ может быть сделан из любой загруженной системы. Если вы решите это сделать, вам нужно будет примонтировать отдельный раздел, переместить все файлы из текущего каталога `/boot` (например, ядро Linux, которое вы только что создали в предыдущем разделе), в новый раздел. Затем вам нужно размонтировать раздел и примонтировать заново в каталог `/boot` . Если вы это сделаете, обязательно обновите данные в файле `/etc/fstab` .

Использование текущего раздела lfs будет работать, но настройка для нескольких систем сложнее.

Используя приведенную выше информацию, определите соответствующие точки монтирования для корневого раздела (или загрузочного раздела, если используется отдельный). В следующем примере предполагается, что корневой (или отдельный загрузочный) раздел является `sda2`.

Установите файлы GRUB в каталог`/boot/grub` и настройте загрузочный сектор:

Warning

Следующая команда перезапишет текущий загрузчик. Не выполняйте эту команду, если это нежелательно, например, если вы используете сторонний менеджер загрузки для управления главной загрузочной записью (MBR).

```
grub-install /dev/sda
```

Note

Если система была загружена с использованием UEFI, **grub-install** попытается установить файлы для *x86_64-efi* , но эти файлы не были кстановлены в главе 6. В этом случае, необходимо добавить аргумент `--target i386-pc` к вышеуказанной комманде.

8.4.4. Создание файла конфигурации GRUB

Создайте файл конфигурации `/boot/grub/grub.cfg` :

```
cat > /boot/grub/grub.cfg << "EOF"
# Begin /boot/grub/grub.cfg
set default=0
set timeout=5

insmod ext2
set root=(hd0,2)

menuentry "GNU/Linux, Linux 5.2.8-lfs-9.0" {
        linux   /boot/vmlinuz-5.2.8-lfs-9.0 root=/dev/sda2 ro
}
EOF
```

Note

С точки зрения GRUB, файлы ядра относительны от используемого раздела. Если вы используете отдельные /boot раздел, удалите /boot из строки linux. Вам также потребуется изменить строку *set root*, чтобы указать на загрузочный раздел.

GRUB - чрезвычайно мощная программа, и она обеспечивает огромное количество вариантов загрузки с самых разных устройств, работающих систем и типов разделов. Существует также множество вариантов настройки таких как графические экраны заставок, звуки воспроизведения, ввод мыши и т. д. детали этих вариантов выходят за рамки этого раздела.

Caution

Существует команда grub-mkconfig, которая может автоматически записывать файл конфигурации. Она использует набор скриптов в файле /etc/grub.d/ и уничтожит любые сделанные вами настройки. Эти сценарии 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.

Chapter 9. Заключение

9.1. Заключение

Отлично! Новая система LFS установлена! Желаем успехов в работе с вашей новой, блестящей, самостоятельно созданной Linux системой.

Хорошая идея создать файл `/etc/lfs-release` . При наличии такого файла, вы будете знать об установленной версии LFS. Если понадобится помощь, вы сможете указать соответствующий номер версий. Создайте такой файл:

```
echo 9.0 > /etc/lfs-release
```

Также неплохо создать файл, которые покажет статус вашей новой системы по отношению к стандартам LSB. Чтобы создать этот файл, запустите:

```
cat > /etc/lsb-release << "EOF"
DISTRIB_ID="Linux From Scratch"
DISTRIB_RELEASE="9.0"
DISTRIB_CODENAME="<your name here>"
DISTRIB_DESCRIPTION="Linux From Scratch"
EOF
```

Не забудьте указать в поле 'DISTRIB_CODENAME' уникальное название вашей новой системы, на ваше усмотрение.

9.2. Вступите в ряды пользователей LFS

Теперь, когда вы закончили изучение книги LFS, и хотите вступить в ряды пользователей LFS, перейдите по ссылке *http://linuxfromscratch.org/cgi-bin/lfscounter.php* и зарегистрируйтесь. Введите ваше имя и версию LFS которую вы использовали.

Давайте выполним перезагрузку в систему LFS.

9.3. Перезагрузка системы

Теперь, когда все программное обеспечение установлено, пришло время перезагрузить компьютер. Однако вы должны знать о нескольких вещах. Система, которую вы создали в этой книге, весьма минималистична и, скорее всего, в ней не будет функциональности, которая вам нужна. Установив некотороые дополнительные пакеты из книги BLFS, пока вы ещё остаетесь в chroot окружении, вы сможете расширить функциональность системы, и продолжить работу после перезагрузки в новую систему LFS. Вот некоторые предложения:

- Текстовый браузер, например, *Lynx* позволит легко изучать книгу BLFS в окне виртуального терминала, по мере сборки пакетов в другом.
- Пакет *GPM* позволит работать с буфером обмена в виртуальных терминалах.
- Если настройка статического IP адреса не соответствует вашим сетевым требованиям, установка пакета *dhcpcd* или его клиентской части *dhcp* может быть полезным.
- Установка пакета *sudo* может быть полезна для сборки пакетов от непревилегированного пользователя и более упрощенной установки пакетов в систему.

- Если вам необходим доступ к новой системе удалённо, с удобным интерфейсом, установите пакет*openssh*.
- Для загрузки файлов через сеть, установите пакет *wget*.
- Если один или несколько дисков имеют разделы GUID (GPT), пакет *gptfdisk* или *parted* будет полезным.
- Наконец, обзор следующих файлов конфигурации будет уместным на этом этапе.
  - /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 наконец перейдём к процессу загрузки нашей новой, созданной системы LFS. Для начала, выйдем из окружения chroot:

```
logout
```

отмонтируйте виртуальный файловые системы:

```
umount -v $LFS/dev/pts
umount -v $LFS/dev
umount -v $LFS/run
umount -v $LFS/proc
umount -v $LFS/sys
```

отмонтируйте файловую систему LFS:

```
umount -v $LFS
```

Если было созданно несколько разделов, отмонтируйте их прежде, чем выполните размонтирование основного. Примерно так:

```
umount -v $LFS/usr
umount -v $LFS/home
umount -v $LFS
```

Теперь, выполним перезагрузку системы:

```
shutdown -r now
```

Предполагая, что загрузчик GRUB был настроен, как описано ранее, пункт меню *LFS 9.0* будет настроен для загрузки системы.

По завершении перезагрузки, система LFS готова к использованию и необходимое программное обеспечение может быть добавлено на ваше усмотрение.

9.4. Что теперь?

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 была установлена, вам может быть интересно “Что дальше?” Чтобы ответить на этот вопрос, мы составили список ресурсов для вас.

- Поддержка

  Bugs and security notices are reported regularly for all software. Since an LFS system is compiled from source, it is up to you to keep abreast of such reports. There are several online resources that track such reports, some of which are shown below:

  - *CERT* (Computer Emergency Response Team)

    CERT has a mailing list that publishes security alerts concerning various operating systems and applications. Subscription information is available at *http://www.us-cert.gov/cas/signup.html*.

  - Bugtraq

    Bugtraq is a full-disclosure computer security mailing list. It publishes newly discovered security issues, and occasionally potential fixes for them. Subscription information is available at *http://www.securityfocus.com/archive*.

- Beyond Linux From Scratch

  The Beyond Linux From Scratch book covers installation procedures for a wide range of software beyond the scope of the LFS Book. The BLFS project is located at *https://linuxfromscratch.ru/blfs/*.

- LFS Hints

  The LFS Hints are a collection of educational documents submitted by volunteers in the LFS community. The hints are available at *https://linuxfromscratch.ru/hints/list.html*.

- Mailing lists

  There are several LFS mailing lists you may subscribe to if you are in need of help, want to stay current with the latest developments, want to contribute to the project, and more. See Chapter 1 - Mailing Lists for more information.

- The Linux Documentation Project

  The goal of The Linux Documentation Project (TLDP) is to collaborate on all of the issues of Linux documentation. The TLDP features a large collection of HOWTOs, guides, and man pages. It is located at *http://www.tldp.org/*.

Part IV. Приложения

Appendix A. Сокращения и условные обозначения

ABI Application Binary Interface (Двоичный (бинарный) интерфейс приложений)

ALFS Automated Linux From Scratch (Автоматизированный Linux From Scratch)

API Application Programming Interface (программный интерфейс приложения, интерфейс прикладного программирования)

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 (за пределами Linux From Scratch)

BSD Berkeley Software Distribution (система распространения программного обеспечения)

chroot change root (операция 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 / центра́льное проце́ссорное устро́йство)

CRC Cyclic Redundancy Check (циклический избыточный код, алгоритм нахождения контрольной суммы)

CVS Concurrent Versions System (система одновременных версий)

DHCP Dynamic Host Configuration Protocol (протокол динамической настройки узла)

DNS Domain Name Service (служба /система доменных имён)

EGA Enhanced Graphics Adapter (усовершенствованный графический адаптер)

ELF Executable and Linkable Format (формат исполнимых и компонуемых файлов)

EOF End of File (конец файла)

EQN equation (уравнение / языка описания математических выражений)

ext2 second extended file system (вторая расширенная файловая система (ФС))

ext3 third extended file system (третья расширенная файловая система (ФС))

ext4 fourth extended file system (четвёртая расширенная файловая система (ФС))

FAQ Frequently Asked Questions (часто задаваемые вопросы)

FHS Filesystem Hierarchy Standard (стандарт иерархии файловой системы)

FIFO First-In, First Out (первым пришёл — первым ушёл)

FQDN Fully Qualified Domain Name (полностью определённое имя домена / полное имя домена)

FTP File Transfer Protocol (протокол передачи файлов)

GB Gigabytes (Гигабайты)

GCC GNU Compiler Collection (коллекция компиляторов проекта 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 (параллельный интерфейс подключения накопителей)

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 (протокол прикладного уровня для обмена сообщениями в режиме реального времени.)

ISO International Organization for Standardization (международная организация по стандартизации)

ISP Internet Service Provider (Интернет-провайдер)

KB Kilobytes (килобайты)

LED Light Emitting Diode (светодиод)

LFS Linux From Scratch

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 (сетевая карта, сетевой адаптер)

NLS Native Language Support (поддержка нативных языков)

NNTP Network News Transport Protocol (сетевой протокол распространения, запрашивания, размещения и получения групп новостей при взаимодействии между сервером групп новостей и клиентом)

NPTL Native POSIX Threading Library (нативная библиотека потоков POSIX)

OSS Open Sound System (унифицированный драйвер для звуковых карт и других звуковых устройств)

PCH Pre-Compiled Headers (предварительно скомпилированые заголовки)

PCRE Perl Compatible Regular Expression (Perl-совместимое регулярное выражение)

PID Process Identifier (идентификатор процесса)

PTY pseudo terminal (псевдотерминал)

QOS Quality Of Service (качество обслуживания / сервиса)
RAM Random Access Memory (запоминающее устройство с произвольным доступом)
RPC Remote Procedure Call (удаленный вызов процедур)
RTC Real Time Clock (часы реального времени)
SBU Standard Build Unit (стандартная единица сборки)
SCO The Santa Cruz Operation
SHA1 Secure-Hash Algorithm 1 (алгоритм криптографического хеширования)
TLDP The Linux Documentation Project (проект документации Linux)
TFTP Trivial File Transfer Protocol (простой протокол передачи файлов)
TLS Thread-Local Storage (потоковое хранилище)
UID User Identifier (идентификатор пользователя)
umask user file-creation mask (маска режима 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)

Appendix B. Благодарности

Мы хотели бы поблагодарить следующих людей и организации за вклад в проект Linux From Scratch.

- *Gerard Beekmans* <gerard@linuxfromscratch.org> – LFS Creator
- *Bruce Dubbs* <bdubbs@linuxfromscratch.org> – LFS Managing Editor
- *Jim Gifford* <jim@linuxfromscratch.org> – CLFS Project Co-Leader
- *Pierre Labastie* <pierre@linuxfromscratch.org> – BLFS Editor and ALFS Lead
- *DJ Lucas* <dj@linuxfromscratch.org> – LFS and BLFS Editor
- *Ken Moffat* <ken@linuxfromscratch.org> – BLFS Editor
- Многие другие люди в списках рассылки LFS и BLFS которые помогли сделать эту книгу, предоставив свои предложения, тестирование книги, а также отправки отчетов об ошибках, инструкций и опыт установки различных пакетов.

Переводчики

- *Manuel Canales Esparcia* <macana@macana-es.com> – Spanish LFS translation project
- *Johan Lenglet* <johan@linuxfromscratch.org> – French LFS translation project until 2008
- *Jean-Philippe Mengual* <jmengual@linuxfromscratch.org> – French LFS translation project 2008-2016
- *Julien Lepiller* <jlepiller@linuxfromscratch.org> – French LFS translation project 2017-present
- *Anderson Lizardo* <lizardo@linuxfromscratch.org> – Portuguese LFS translation project
- *Thomas Reitelbach* <tr@erdfunkstelle.de> – German LFS translation project
- *Anton Maisak* <info@linuxfromscratch.ru> – Russian LFS translation project
- *Elena Shevcova* <helen@linuxfromscratch.ru> – Russian LFS translation project

Сопровождающие зеркал сайтов проекта

Северная Америка

- *Scott Kveton* <scott@osuosl.org> – lfs.oregonstate.edu mirror
- *William Astle* <lost@l-w.net> – ca.linuxfromscratch.org mirror
- *Eujon Sellers* <jpolen@rackspace.com> – lfs.introspeed.com mirror
- *Justin Knierim* <tim@idge.net> – lfs-matrix.net mirror

Южная Америка

- *Manuel Canales Esparcia* <manuel@linuxfromscratch.org> – lfsmirror.lfs-es.info mirror
- *Luis Falcon* <Luis Falcon> – torredehanoi.org mirror

Европа

- *Guido Passet* <guido@primerelay.net> – nl.linuxfromscratch.org mirror

- *Bastiaan Jacques* <baafie@planet.nl> – lfs.pagefault.net mirror
- *Sven Cranshoff* <sven.cranshoff@lineo.be> – lfs.lineo.be mirror
- Scarlet Belgium – lfs.scarlet.be mirror
- *Sebastian Faulborn* <info@aliensoft.org> – lfs.aliensoft.org mirror
- *Stuart Fox* <stuart@dontuse.ms> – lfs.dontuse.ms mirror
- *Ralf Uhlemann* <admin@realhost.de> – lfs.oss-mirror.org mirror
- *Antonin Sprinzl* <Antonin.Sprinzl@tuwien.ac.at> – at.linuxfromscratch.org mirror
- *Fredrik Danerklint* <fredan-lfs@fredan.org> – se.linuxfromscratch.org mirror
- *Franck* <franck@linuxpourtous.com> – lfs.linuxpourtous.com mirror
- *Philippe Baque* <baque@cict.fr> – lfs.cict.fr mirror
- *Benjamin Heil* <kontakt@wankoo.org> – lfs.wankoo.org mirror

Россия

- *Vitaly Chekasin* <gyouja@pilgrims.ru> – lfs.pilgrims.ru mirror
- *Anton Maisak* <info@linuxfromscratch.ru> – linuxfromscratch.ru mirror

Азия

- *Satit Phermsawang* <satit@wbac.ac.th> – lfs.phayoune.org mirror
- *Shizunet Co.,Ltd.* <info@shizu-net.jp> – lfs.mirror.shizu-net.jp mirror
- *Init World* <http://www.initworld.com/> – lfs.initworld.com mirror

Австралия

- *Jason Andrade* <jason@dstc.edu.au> – au.linuxfromscratch.org mirror

Бывшие члены команды проекта

- *Christine Barczak* <theladyskye@linuxfromscratch.org> – LFS Book Editor
- Archaic <archaic@linuxfromscratch.org> – LFS Technical Writer/Editor, HLFS Project Leader, BLFS Editor, Hints and Patches Project Maintainer
- *Matthew Burgess* <matthew@linuxfromscratch.org> – LFS Project Leader, LFS Technical Writer/Editor
- *Nathan Coulson* <nathan@linuxfromscratch.org> – LFS-Bootscripts Maintainer
- Timothy Bauscher
- Robert Briggs
- Ian Chilton
- *Jeroen Coumans* <jeroen@linuxfromscratch.org> – Website Developer, FAQ Maintainer
- *Manuel Canales Esparcia* <manuel@linuxfromscratch.org> – LFS/BLFS/HLFS XML and XSL Maintainer
- Alex Groenewoud – LFS Technical Writer
- Marc Heerdink

- *Jeremy Huntwork* <jhuntwork@linuxfromscratch.org> – LFS Technical Writer, LFS LiveCD Maintainer
- *Bryan Kadzban* <bryan@linuxfromscratch.org> – LFS Technical Writer
- Mark Hymers
- Seth W. Klein – FAQ maintainer
- *Nicholas Leippe* <nicholas@linuxfromscratch.org> – Wiki Maintainer
- *Anderson Lizardo* <lizardo@linuxfromscratch.org> – Website Backend-Scripts Maintainer
- *Randy McMurchy* <randy@linuxfromscratch.org> – BLFS Project Leader, LFS Editor
- *Dan Nicholson* <dnicholson@linuxfromscratch.org> – LFS and BLFS Editor
- *Alexander E. Patrakov* <alexander@linuxfromscratch.org> – LFS Technical Writer, LFS Internationalization Editor, LFS Live CD Maintainer
- Simon Perreault
- *Scot Mc Pherson* <scot@linuxfromscratch.org> – LFS NNTP Gateway Maintainer
- *Douglas R. Reno* <renodr@linuxfromscratch.org> – Systemd Editor
- *Ryan Oliver* <ryan@linuxfromscratch.org> – CLFS Project Co-Leader
- *Greg Schafer* <gschafer@zip.com.au> – LFS Technical Writer and Architect of the Next Generation 64-bit-enabling Build Method
- Jesse Tie-Ten-Quee – LFS Technical Writer
- *James Robertson* <jwrober@linuxfromscratch.org> – Bugzilla Maintainer
- *Tushar Teredesai* <tushar@linuxfromscratch.org> – BLFS Book Editor, Hints and Patches Project Leader
- *Jeremy Utley* <jeremy@linuxfromscratch.org> – LFS Technical Writer, Bugzilla Maintainer, LFS-Bootscripts Maintainer
- *Zack Winkles* <zwinkles@gmail.com> – LFS Technical Writer

Appendix C. Зависимости

Каждый пакет в системе LFS может ссылаться на один или несколько других пакетов в определённом порядке, для того чтобы сборка и установка пакета была выполнена правильно. Некоторые пакеты могут иметь циклическую зависимость, 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 заключается в описании зависимостей каждого пакета

Для каждого пакета, будет перечислено три, а иногда даже четыре типа зависимостей. Первый список перечисляет какие пакеты должны быть, для компиляции и установки текущего пакета. Второй список перечисляет какие пакеты, помимо тех, которые указаны в первом списке, должны присутствовать, чтобы запустить наборы тестов. Третий список перечисляет какие пакеты требуют, чтобы указанный пакет был собран и установлен до их сборки и установки. В большинстве случаев это происходит потому, что скрипты конфигурации пакетов жестко прописывают ссылки на другие файлы. Если выполнять сборку не в требуемом порядке, это можем привести к тому, что результаты действий будут сохраняться по пути /tools/bin/[binary], что само собой не желательно.

Последний список зависимостей - это опциональные пакеты, которые могут не упоминаться в книге LFS, но все же, могут быть полезными для пользователя. 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

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, Grep, M4, Make, Perl, Sed, и Texinfo

Набор тестов зависит от: Automake, Diffutils, Findutils, и Libtool

Должен быть установлен до: Acl and Libcap

Необязательные зависимости: Нет

Autoconf

Установка зависит от: Bash, Coreutils, Grep, M4, Make, Perl, Sed, и Texinfo

Набор тестов зависит от: Automake, Diffutils, Findutils, GCC, и Libtool

Должен быть установлен до: Automake

Необязательные зависимости: Emacs

Automake

Установка зависит от: Autoconf, Bash, Coreutils, Gettext, Grep, M4, Make, Perl, 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: Нет

Необязательные зависимости: Нет

Bash

Установка зависит от: Bash, Binutils, Bison, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Make, Ncurses, Patch, Readline, Sed, и Texinfo

Набор тестов зависит от: Shadow

Должен быть установлен до: Нет

Необязательные зависимости: Xorg

Bc

Установка зависит от: Bash, Binutils, Bison, Coreutils, GCC, Glibc, Grep, Make, Perl, и Readline

Набор тестов зависит от: Gawk

Должен быть установлен до: Linux Kernel

Необязательные зависимости: Нет

Binutils

Установка зависит от: Bash, Binutils, Coreutils, Diffutils, File, Gawk, GCC, Glibc, Grep, Make, Perl, Sed, Texinfo and Zlib

Набор тестов зависит от: DejaGNU and Expect

Должен быть установлен до: Нет

Необязательные зависимости: Нет

Bison

Установка зависит от: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, M4, Make, Perl, и Sed

Набор тестов зависит от: Diffutils, Findutils, и Flex

Должен быть установлен до: Kbd and Tar

Необязательные зависимости: Doxygen (test suite)

Bzip2

Установка зависит от: Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Make, and Patch

Набор тестов зависит от: Нет

Должен быть установлен до: Нет

Необязательные зависимости: Нет

Check

Установка зависит от: GCC, Grep, Make, Sed, и Texinfo

Набор тестов зависит от: Нет

Должен быть установлен до: Нет

Необязательные зависимости: Нет

Coreutils

Установка зависит от: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, GMP, Grep, Make, Patch, Perl, Sed, и Texinfo

Набор тестов зависит от: Diffutils, E2fsprogs, Findutils, Shadow, и Util-linux

Должен быть установлен до: Bash, Diffutils, Eudev, Findutils, и Man-DB

Необязательные зависимости: Perl Expect and IO:Tty modules (for test suite)

DejaGNU

Установка зависит от: Bash, Coreutils, Diffutils, GCC, Grep, Make, и Sed

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

Набор тестов зависит от: Perl

Должен быть установлен до: Нет

Необязательные зависимости: Нет

Eudev

Установка зависит от: Bash, Binutils, Coreutils, Gawk, GCC, Glibc, Grep, Gperf, Make, и Sed

Набор тестов зависит от: Нет

Должен быть установлен до: Нет

Необязательные зависимости: Нет

Expat

Установка зависит от: Bash, Binutils, Coreutils, Gawk, GCC, Glibc, Grep, Make, и Sed

Набор тестов зависит от: Нет

Должен быть установлен до: XML::Parser

Необязательные зависимости: Нет

Expect

Установка зависит от: Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Grep, Make, Patch, Sed, и Tcl

Набор тестов зависит от: Нет

Должен быть установлен до: Нет

Необязательные зависимости: Нет

E2fsprogs

Установка зависит от: Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Gzip, Make, Sed, Texinfo, и Util-linux

Набор тестов зависит от: Procps-ng and Psmisc

Должен быть установлен до: Нет

Необязательные зависимости: Нет

File

Установка зависит от: Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Make, Sed, и Zlib

Набор тестов зависит от: Нет

Должен быть установлен до: Нет

Необязательные зависимости: Нет

Findutils

Установка зависит от: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, Make, Sed, и Texinfo

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

Набор тестов зависит от: Bison and Gawk

Должен быть установлен до: IPRoute2, Kbd, и Man-DB

Необязательные зависимости: Нет

Gawk

Установка зависит от: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, GMP, Grep, Make, MPFR, Patch, Readline, Sed, и Texinfo

Набор тестов зависит от: Diffutils

Должен быть установлен до: Нет

Необязательные зависимости: Нет

Gcc

Установка зависит от: Bash, Binutils, Coreutils, Diffutils, Findutils, Gawk, GCC, Gettext, Glibc, GMP, Grep, M4, Make, MPC, MPFR, Patch, Perl, Sed, Tar, и Texinfo

Набор тестов зависит от: DejaGNU, Expect, и Shadow

Должен быть установлен до: Нет

Необязательные зависимости: *GNAT* and *ISL*

GDBM

Установка зависит от: Bash, Binutils, Coreutils, Diffutils, GCC, Grep, Make, и Sed

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, Sed, и Texinfo

Набор тестов зависит от: Diffutils, Perl, и Tcl

Должен быть установлен до: Automake

Необязательные зависимости: Нет

Glibc

Установка зависит от: Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Gettext, Grep, Gzip, Linux API Headers, Make, Perl, Python, Sed, и Texinfo

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

Набор тестов зависит от: Нет

Должен быть установлен до: MPFR and GCC

Необязательные зависимости: Нет

Gperf

Установка зависит от: Bash, Binutils, Coreutils, GCC, Glibc, и Make

Набор тестов зависит от: Diffutils and 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

Набор тестов зависит от: Gawk

Должен быть установлен до: Man-DB

Необязательные зависимости: Pcre

Groff

Установка зависит от: Bash, Binutils, Bison, Coreutils, Gawk, GCC, Glibc, Grep, Make, Patch, Sed, и Texinfo

Набор тестов зависит от: Тесты недоступны

Должен быть установлен до: Man-DB and Perl

Необязательные зависимости: GPL Ghostscript

GRUB

Установка зависит от: Bash, Binutils, Bison, Coreutils, Diffutils, GCC, Gettext, Glibc, Grep, Make, Ncurses, Sed, Texinfo, и Xz

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

Набор тестов зависит от: Diffutils and Less

Должен быть установлен до: Man-DB

Необязательные зависимости: Нет

Iana-Etc

Установка зависит от: Coreutils, Gawk, и Make

Набор тестов зависит от: Тесты недоступны

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

Набор тестов зависит от: Тесты недоступны

Должен быть установлен до: Tar

Необязательные зависимости: Нет

Intltool

Установка зависит от: Bash, Gawk, Glibc, Make, Perl, Sed, и XML::Parser

Набор тестов зависит от: Perl

Должен быть установлен до: Нет

Необязательные зависимости: Нет

IProute2

Установка зависит от: Bash, Bison, Coreutils, Flex, GCC, Glibc, Make, and Linux API Headers

Набор тестов зависит от: Тесты недоступны

Должен быть установлен до: Нет

Необязательные зависимости: Нет

Kbd

Установка зависит от: Bash, Binutils, Bison, Check, Coreutils, Flex, GCC, Gettext, Glibc, Gzip, Make, Patch, и Sed

Набор тестов зависит от: Тесты недоступны

Должен быть установлен до: Нет

Необязательные зависимости: Нет

Kmod

Установка зависит от: Bash, Binutils, Bison, Coreutils, Flex, GCC, Gettext, Glibc, Gzip, Make, Pkg-config, Sed, Xz-Utils, and Zlib

Набор тестов зависит от: Тесты недоступны

Должен быть установлен до: Eudev

Необязательные зависимости: Нет

Less

Установка зависит от: Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Grep, Make, Ncurses, и Sed

Набор тестов зависит от: Тесты недоступны

Должен быть установлен до: Gzip

Необязательные зависимости: Pcre

Libcap

Установка зависит от: Attr, Bash, Binutils, Coreutils, GCC, Glibc, Perl, Make, and Sed

Набор тестов зависит от: Тесты недоступны

Должен быть установлен до: Нет

Необязательные зависимости: Linux-PAM

Libelf

Установка зависит от: Bash, Binutils, Coreutils, GCC, Glibc, и Make

Набор тестов зависит от: No test suite available

Должен быть установлен до: Linux Kernel

Необязательные зависимости: None

Libffi

Установка зависит от: Bash, Binutils, Coreutils, GCC, Glibc, Make, и Sed

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

Набор тестов зависит от: Check

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

Набор тестов зависит от: Autoconf, Automake, и Findutils

Должен быть установлен до: Нет

Необязательные зависимости: Нет

Linux Kernel

Установка зависит от: Bash, Bc, Binutils, Coreutils, Diffutils, Findutils, GCC, Glibc, Grep, Gzip, Kmod, Make, Ncurses, OpenSSL, Perl, и Sed

Набор тестов зависит от: Тесты недоступны

Должен быть установлен до: Нет

Необязательные зависимости: Нет

M4

Установка зависит от: Bash, Binutils, Coreutils, GCC, Glibc, Grep, Make, Sed, and Texinfo

Набор тестов зависит от: Diffutils

Должен быть установлен до: Autoconf and Bison

Необязательные зависимости: libsigsegv

Make

Установка зависит от: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, Make, Sed, и Texinfo

Набор тестов зависит от: Perl and Procps-ng

Должен быть установлен до: Нет

Необязательные зависимости: Нет

Man-DB

Установка зависит от: Bash, Binutils, Bzip2, Coreutils, Flex, GCC, GDBM, Gettext, Glibc, Grep, Groff, Gzip, Less, Libpipeline, Make, Sed, и Xz

Набор тестов зависит от: Util-linux

Должен быть установлен до: Нет

Необязательные зависимости: Нет

Man-Pages

Установка зависит от: Bash, Coreutils, и Make

Набор тестов зависит от: Тесты недоступны

Должен быть установлен до: Нет

Необязательные зависимости: Нет

Meson

Установка зависит от: Ninja and Python

Набор тестов зависит от: Тесты недоступны

Должен быть установлен до: Systemd

Необязательные зависимости: Нет

MPC

Установка зависит от: Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, GMP, Make, MPFR, Sed, и Texinfo

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

Набор тестов зависит от: Нет

Должен быть установлен до: Gawk and GCC

Необязательные зависимости: Нет

Ncurses

Установка зависит от: Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Make, Patch, и Sed

Набор тестов зависит от: Тесты недоступны

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

Набор тестов зависит от: Нет

Должен быть установлен до: Meson

Необязательные зависимости: Asciidoc, Doxygen, Emacs, и re2c

Openssl

Установка зависит от: Binutils, Coreutils, Gcc, Make, и Perl

Набор тестов зависит от: Нет

Должен быть установлен до: Linux

Необязательные зависимости: Нет

Patch

Установка зависит от: Bash, Binutils, Coreutils, GCC, Glibc, Grep, Make, и Sed

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, Groff, Make, Sed, и Zlib

Набор тестов зависит от: Iana-Etc and Procps-ng

Должен быть установлен до: Autoconf

Необязательные зависимости: Нет

Pkg-config

Установка зависит от: Bash, Binutils, Coreutils, Gawk, GCC, Glibc, Grep, Make, Popt, и Sed

Набор тестов зависит от: Нет

Должен быть установлен до: Kmod

Необязательные зависимости: Нет

Popt

Установка зависит от: Bash, Binutils, Coreutils, Gawk, GCC, Glibc, Grep, и Make

Набор тестов зависит от: Diffutils and Sed

Должен быть установлен до: Pkg-config

Необязательные зависимости: Нет

Procps-ng

Установка зависит от: Bash, Binutils, Coreutils, GCC, Glibc, Make, и Ncurses

Набор тестов зависит от: DejaGNU

Должен быть установлен до: Нет

Необязательные зависимости: Нет

Psmisc

Установка зависит от: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, Make, Ncurses, и Sed

Набор тестов зависит от: Тесты недоступны

Должен быть установлен до: Нет

Необязательные зависимости: Нет

Python

Установка зависит от: Bash, Binutils, Coreutils, GCC, Gdbm, Gettext, Glibc, Grep, Libffi, Make, Ncurses, и Sed

Набор тестов зависит от: GDB and Valgrind

Должен быть установлен до: Ninja

Необязательные зависимости: Berkeley DB, OpenSSL, SQLite, и Tk

Readline

Установка зависит от: Bash, Binutils, Coreutils, Gawk, GCC, Glibc, Grep, Make, Ncurses, Patch, Sed, и Texinfo

Набор тестов зависит от: Тесты недоступны

Должен быть установлен до: Bash and Gawk

Необязательные зависимости: Нет

Sed

Установка зависит от: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, Make, Sed, и Texinfo

Набор тестов зависит от: Diffutils and Gawk

Должен быть установлен до: E2fsprogs, File, Libtool, и Shadow

Необязательные зависимости: Нет

Shadow

Установка зависит от: Bash, Binutils, Coreutils, Diffutils, Findutils, Gawk, GCC, Gettext, Glibc, Grep, Make, и Sed

Набор тестов зависит от: Тесты недоступны

Должен быть установлен до: Coreutils

Необязательные зависимости: Acl, Attr, Cracklib, и PAM

Sysklogd

Установка зависит от: Binutils, Coreutils, GCC, Glibc, Make, и Patch

Набор тестов зависит от: Тесты недоступны

Должен быть установлен до: Нет

Необязательные зависимости: Нет

Systemd

Установка зависит от: Acl, Attr, Bash, Binutils, Coreutils, Diffutils, Expat, Gawk, GCC, Glibc, Gperf, Grep, Intltool, Libcap, Meson, Sed, and Util-linux

Набор тестов зависит от: Нет

Должен быть установлен до: Нет

Необязательные зависимости: Много, подробнее - *BLFS systemd page*

Sysvinit

Установка зависит от: Binutils, Coreutils, GCC, Glibc, Make, и Sed

Набор тестов зависит от: Тесты недоступны

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

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

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

Набор тестов зависит от: Нет

Должен быть установлен до: Нет

Необязательные зависимости: Нет

Util-linux

Установка зависит от: Bash, Binutils, Coreutils, Diffutils, Eudev, Findutils, Gawk, GCC, Gettext, Glibc, Grep, Make, Ncurses, Sed, и Zlib

Набор тестов зависит от: Нет

Должен быть установлен до: Нет

Необязательные зависимости: *libcap-ng*

Vim

Установка зависит от: Acl, Attr, Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Grep, Make, Ncurses, и Sed

Набор тестов зависит от: Нет

Должен быть установлен до: Нет

Необязательные зависимости: Xorg, GTK+2, LessTif, Python, Tcl, Ruby, и GPM

XML::Parser

Установка зависит от: Bash, Binutils, Coreutils, Expat, GCC, Glibc, Make, 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

Набор тестов зависит от: Нет

Должен быть установлен до: Eudev, GRUB, Kmod, и Man-DB

Необязательные зависимости: Нет

Zlib

Установка зависит от:	Bash, Binutils, Coreutils, GCC, Glibc, Make, и Sed
Набор тестов зависит от:	Нет
Должен быть установлен до:	File, Kmod, Perl, и Util-linux
Необязательные зависимости:	Нет

Appendix D. Скрипты загрузки и системной конфигурации версия-20190524

Скрипты в этом приложении перечислены с указанием места, где как правило, они должны находится. `/etc/rc.d/init.d`, `/etc/sysconfig`, `/etc/sysconfig/network-devices`, and `/etc/sysconfig/network-devices/services`. Каждый раздел перечислен в том порядке, в котором как правило происходит вызов

D.1. /etc/rc.d/init.d/rc

`rc` скрипт является первым вызываемым скриптом системой инициализации init и предназначен для выполнения процесса загрузки.

```
#!/bin/bash
#############################################################################
# Begin rc
#
# Description : Main Run Level Control Script
#
# Authors     : Gerard Beekmans  - gerard@linuxfromscratch.org
#             : DJ Lucas - dj@linuxfromscratch.org
# Update      : Bruce Dubbs - bdubbs@linuxfromscratch.org
#
# Version     : LFS 7.0
#
#############################################################################

. /lib/lsb/init-functions

print_error_msg()
{
   log_failure_msg
   # $i is set when called
   MSG="FAILURE:\n\nYou should not be reading this error message.\n\n"
   MSG="${MSG}It means that an unforeseen error took place in\n"
   MSG="${MSG}${i},\n"
   MSG="${MSG}which exited with a return value of ${error_value}.\n"

   MSG="${MSG}If you're able to track this error down to a bug in one of\n"
   MSG="${MSG}the files provided by the ${DISTRO_MINI} book,\n"
   MSG="${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@linuxfromscratch.org (Registration required)"}
DISTRO_MINI=${DISTRO_MINI:-"LFS"}
IPROMPT=${IPROMPT:-"no"}

# These 3 signals will not cause our script to exit
trap "" INT QUIT TSTP

[ "${1}" != "" ] && runlevel=${1}

if [ "${runlevel}" == "" ]; then
   echo "Usage: ${0} <runlevel>" >&2
```

```
   exit 1
fi

previous=${PREVLEVEL}
[ "${previous}" == "" ] && previous=N

if [ ! -d /etc/rc.d/rc${runlevel}.d ]; then
   log_info_msg "/etc/rc.d/rc${runlevel}.d does not exist.\n"
   exit 1
fi

if [ "$runlevel" == "6" -o "$runlevel" == "0" ]; then IPROMPT="no"; fi

# Note: In ${LOGLEVEL:-7}, it is ':' 'dash' '7', not minus 7
if [ "$runlevel" == "S" ]; then
   [ -r /etc/sysconfig/console ] && source /etc/sysconfig/console
   dmesg -n "${LOGLEVEL:-7}"
fi

if [ "${IPROMPT}" == "yes" -a "${runlevel}" == "S" ]; then
   # The total length of the distro welcome string, without escape codes
   wlen=${wlen:-$(echo "Welcome to ${DISTRO}" | wc -c )}
   welcome_message=${welcome_message:-"Welcome to ${INFO}${DISTRO}${NORMAL}"}

   # The total length of the interactive string, without escape codes
   ilen=${ilen:-$(echo "Press 'I' to enter interactive startup" | wc -c )}
   i_message=${i_message:-"Press '${FAILURE}I${NORMAL}' to enter interactive startup"}


   # dcol and icol are spaces before the message to center the message
   # on screen. itime is the amount of wait time for the user to press a key
   wcol=$(( ( ${COLUMNS} - ${wlen} ) / 2 ))
   icol=$(( ( ${COLUMNS} - ${ilen} ) / 2 ))
   itime=${itime:-"3"}

   echo -e "\n\n"
   echo -e "\\033[${wcol}G${welcome_message}"
   echo -e "\\033[${icol}G${i_message}${NORMAL}"
   echo ""
   read -t "${itime}" -n 1 interactive 2>&1 > /dev/null
fi

# Make lower case
[ "${interactive}" == "I" ] && interactive="i"
[ "${interactive}" != "i" ] && interactive=""

# Read the state file if it exists from runlevel S
[ -r /var/run/interactive ] && source /var/run/interactive

# Attempt to stop all services started by the previous runlevel,
# and killed in this runlevel
if [ "${previous}" != "N" ]; then
   for i in $(ls -v /etc/rc.d/rc${runlevel}.d/K* 2> /dev/null)
   do
      check_script_status
      if [ "${SCRIPT_STAT}" == "1" ]; then
         SCRIPT_STAT="0"
```

```
         continue
      fi

      suffix=${i#/etc/rc.d/rc$runlevel.d/K[0-9][0-9]}
      prev_start=/etc/rc.d/rc$previous.d/S[0-9][0-9]$suffix
      sysinit_start=/etc/rc.d/rcS.d/S[0-9][0-9]$suffix

      if [ "${runlevel}" != "0" -a "${runlevel}" != "6" ]; then
         if [ ! -f ${prev_start} -a  ! -f ${sysinit_start} ]; then
            MSG="WARNING:\n\n${i} can't be "
            MSG="${MSG}executed because it was not "
            MSG="${MSG}not started in the previous "
            MSG="${MSG}runlevel (${previous})."
            log_warning_msg "$MSG"
            continue
         fi
      fi

      run ${i} stop
      error_value=${?}

      if [ "${error_value}" != "0" ]; then print_error_msg; fi
   done
fi

if [ "${previous}" == "N" ]; then export IN_BOOT=1; fi

if [ "$runlevel" == "6" -a -n "${FASTBOOT}" ]; then
   touch /fastboot
fi


# Start all functions in this runlevel
for i in $( ls -v /etc/rc.d/rc${runlevel}.d/S* 2> /dev/null)
do
   if [ "${previous}" != "N" ]; then
      suffix=${i#/etc/rc.d/rc$runlevel.d/S[0-9][0-9]}
      stop=/etc/rc.d/rc$runlevel.d/K[0-9][0-9]$suffix
      prev_start=/etc/rc.d/rc$previous.d/S[0-9][0-9]$suffix

      [ -f ${prev_start} -a ! -f ${stop} ] && continue
   fi

   check_script_status
      if [ "${SCRIPT_STAT}" == "1" ]; then
         SCRIPT_STAT="0"
         continue
      fi

   case ${runlevel} in
      0|6)
         run ${i} stop
         ;;
      *)
         run ${i} start
         ;;
   esac
```

```
   error_value=${?}

   if [ "${error_value}" != "0" ]; then print_error_msg; fi
done

# Store interactive variable on switch from runlevel S and remove if not
if [ "${runlevel}" == "S" -a "${interactive}" == "i" ]; then
    echo "interactive=\"i\"" > /var/run/interactive
else
    rm -f /var/run/interactive 2> /dev/null
fi

# Copy the boot log on initial boot only
if [ "${previous}" == "N" -a  "${runlevel}" != "S" ]; then
   cat $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

## Screen Dimensions
# Find current screen size
if [ -z "${COLUMNS}" ]; then
   COLUMNS=$(stty size)
   COLUMNS=${COLUMNS##* }
fi

# When using remote connections, such as a serial port, stty size returns 0
if [ "${COLUMNS}" = "0" ]; then
   COLUMNS=80
fi

## Measurements for positioning result messages
COL=$((${COLUMNS} - 8))
WCOL=$((${COL} - 2))

## Set Cursor Position Commands, used via echo
SET_COL="\\033[${COL}G"      # at the $COL char
SET_WCOL="\\033[${WCOL}G"    # at the $WCOL char
CURS_UP="\\033[1A\\033[0G"   # Up one line, at the 0'th char
CURS_ZERO="\\033[0G"

##############################################################################
# start_daemon()                                                             #
# Usage: start_daemon [-f] [-n nicelevel] [-p pidfile] pathname [args...]    #
```

```
#                                                                              #
# Purpose: This runs the specified program as a daemon                         #
#                                                                              #
# Inputs: -f: (force) run the program even if it is already running.           #
#         -n nicelevel: specify a nice level. See 'man nice(1)'.               #
#         -p pidfile: use the specified file to determine PIDs.                #
#         pathname: the complete path to the specified program                 #
#         args: additional arguments passed to the program (pathname)          #
#                                                                              #
# Return values (as defined by LSB exit codes):                                #
#       0 - program is running or service is OK                                #
#       1 - generic or unspecified error                                       #
#       2 - invalid or excessive argument(s)                                   #
#       5 - program is not installed                                           #
################################################################################
start_daemon()
{
    local force=""
    local nice="0"
    local pidfile=""
    local pidlist=""
    local retval=""

    # Process arguments
    while true
    do
        case "${1}" in

            -f)
                force="1"
                shift 1
                ;;

            -n)
                nice="${2}"
                shift 2
                ;;

            -p)
                pidfile="${2}"
                shift 2
                ;;

            -*)
                return 2
                ;;

            *)
                program="${1}"
                break
                ;;
        esac
    done

    # Check for a valid program
    if [ ! -e "${program}" ]; then return 5; fi
```

```
    # Execute
    if [ -z "${force}" ]; then
        if [ -z "${pidfile}" ]; then
            # Determine the pid by discovery
            pidlist=`pidofproc "${1}"`
            retval="${?}"
        else
            # The PID file contains the needed PIDs
            # Note that by LSB requirement, the path must be given to pidofproc,
            # however, it is not used by the current implementation or standard.
            pidlist=`pidofproc -p "${pidfile}" "${1}"`
            retval="${?}"
        fi

        # Return a value ONLY
        # It is the init script's (or distribution's functions) responsibilty
        # to log messages!
        case "${retval}" in

            0)
                # Program is already running correctly, this is a
                # successful start.
                return 0
                ;;

            1)
                # Program is not running, but an invalid pid file exists
                # remove the pid file and continue
                rm -f "${pidfile}"
                ;;

            3)
                # Program is not running and no pidfile exists
                # do nothing here, let start_deamon continue.
                ;;

            *)
                # Others as returned by status values shall not be interpreted
                # and returned as an unspecified error.
                return 1
                ;;
        esac
    fi

    # Do the start!
    nice -n "${nice}" "${@}"
}

########################################################################
# killproc()                                                           #
# Usage: killproc [-p pidfile] pathname [signal]                       #
#                                                                      #
# Purpose: Send control signals to running processes                   #
#                                                                      #
# Inputs: -p pidfile, uses the specified pidfile                       #
#         pathname, pathname to the specified program                  #
#         signal, send this signal to pathname                         #
```

```
#                                                                              #
# Return values (as defined by LSB exit codes):                                #
#       0 - program (pathname) has stopped/is already stopped or a             #
#           running program has been sent specified signal and stopped         #
#           successfully                                                       #
#       1 - generic or unspecified error                                       #
#       2 - invalid or excessive argument(s)                                   #
#       5 - program is not installed                                           #
#       7 - program is not running and a signal was supplied                   #
################################################################################
killproc()
{
    local pidfile
    local program
    local prefix
    local progname
    local signal="-TERM"
    local fallback="-KILL"
    local nosig
    local pidlist
    local retval
    local pid
    local delay="30"
    local piddead
    local dtime

    # Process arguments
    while true; do
        case "${1}" in
            -p)
               pidfile="${2}"
               shift 2
               ;;

             *)
                program="${1}"
                if [ -n "${2}" ]; then
                    signal="${2}"
                    fallback=""
                else
                    nosig=1
                fi

                # Error on additional arguments
                if [ -n "${3}" ]; then
                    return 2
                else
                    break
                fi
                ;;
        esac
    done

    # Check for a valid program
    if [ ! -e "${program}" ]; then return 5; fi

    # Check for a valid signal
```

```
    check_signal "${signal}"
    if [ "${?}" -ne "0" ]; then return 2; fi

    # Get a list of pids
    if [ -z "${pidfile}" ]; then
        # determine the pid by discovery
        pidlist=`pidofproc "${1}"`
        retval="${?}"
    else
        # The PID file contains the needed PIDs
        # Note that by LSB requirement, the path must be given to pidofproc,
        # however, it is not used by the current implementation or standard.
        pidlist=`pidofproc -p "${pidfile}" "${1}"`
        retval="${?}"
    fi

    # Return a value ONLY
    # It is the init script's (or distribution's functions) responsibilty
    # to log messages!
    case "${retval}" in

        0)
            # Program is running correctly
            # Do nothing here, let killproc continue.
            ;;

        1)
            # Program is not running, but an invalid pid file exists
            # Remove the pid file.
            rm -f "${pidfile}"

            # This is only a success if no signal was passed.
            if [ -n "${nosig}" ]; then
                return 0
            else
                return 7
            fi
            ;;

        3)
            # Program is not running and no pidfile exists
            # This is only a success if no signal was passed.
            if [ -n "${nosig}" ]; then
                return 0
            else
                return 7
            fi
            ;;

        *)
            # Others as returned by status values shall not be interpreted
            # and returned as an unspecified error.
            return 1
            ;;
    esac

    # Perform different actions for exit signals and control signals
```

```
    check_sig_type "${signal}"

    if [ "${?}" -eq "0" ]; then # Signal is used to terminate the program

        # Account for empty pidlist (pid file still exists and no
        # signal was given)
        if [ "${pidlist}" != "" ]; then

            # Kill the list of pids
            for pid in ${pidlist}; do

                kill -0 "${pid}" 2> /dev/null

                if [ "${?}" -ne "0" ]; then
                    # Process is dead, continue to next and assume all is well
                    continue
                else
                    kill "${signal}" "${pid}" 2> /dev/null

                    # Wait up to ${delay}/10 seconds to for "${pid}" to
                    # terminate in 10ths of a second

                    while [ "${delay}" -ne "0" ]; do
                        kill -0 "${pid}" 2> /dev/null || piddead="1"
                        if [ "${piddead}" = "1" ]; then break; fi
                        sleep 0.1
                        delay="$(( ${delay} - 1 ))"
                    done

                    # If a fallback is set, and program is still running, then
                    # use the fallback
                    if [ -n "${fallback}" -a "${piddead}" != "1" ]; then
                        kill "${fallback}" "${pid}" 2> /dev/null
                        sleep 1
                        # Check again, and fail if still running
                        kill -0 "${pid}" 2> /dev/null && return 1
                    fi
                fi
            done
        fi

        # Check for and remove stale PID files.
        if [ -z "${pidfile}" ]; then
            # Find the basename of $program
            prefix=`echo "${program}" | sed 's/[^/]*$//'`
            progname=`echo "${program}" | sed "s@${prefix}@@"`

            if [ -e "/var/run/${progname}.pid" ]; then
                rm -f "/var/run/${progname}.pid" 2> /dev/null
            fi
        else
            if [ -e "${pidfile}" ]; then rm -f "${pidfile}" 2> /dev/null; fi
        fi

    # For signals that do not expect a program to exit, simply
    # let kill do 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 "/var/run/${progname}.pid" ]; then
            pidfile="/var/run/${progname}.pid"
        fi
    fi

    # If a PID file is set and exists, use it.
    if [ -n "${pidfile}" -a -e "${pidfile}" ]; then

        # Use the value in the first line of the pidfile
        pidlist=`/bin/head -n1 "${pidfile}"`
        # This can optionally be written as 'sed 1q' to repalce 'head -n1'
        # should LFS move /bin/head to /usr/bin/head
    else
        # Use pidof
        pidlist=`pidof "${program}"`
    fi

    # Figure out if all listed PIDs are running.
    for pid in ${pidlist}; do
        kill -0 ${pid} 2> /dev/null

        if [ "${?}" -eq "0" ]; then
            lpids="${lpids}${pid} "
        else
            exitstatus="1"
        fi
    done

    if [ -z "${lpids}" -a ! -f "${pidfile}" ]; then
        return 3
    else
        echo "${lpids}"
        return "${exitstatus}"
    fi
}

################################################################################
# statusproc()                                                                 #
# Usage: statusproc [-p pidfile] pathname                                      #
#                                                                              #
# Purpose: This function prints the status of a particular daemon to stdout    #
#                                                                              #
# Inputs: -p pidfile, use the specified pidfile instead of pidof               #
#         pathname, path to the specified program                              #
#                                                                              #
# Return values:                                                               #
#       0 - Status printed                                                     #
#       1 - Input error. The daemon to check was not specified.                #
```

```
########################################################################################
statusproc()
{
   local pidfile
   local pidlist

   if [ "${#}" = "0" ]; then
      echo "Usage: statusproc [-p pidfle] {program}"
      exit 1
   fi

   # Process arguments
   while true; do
       case "${1}" in

           -p)
               pidfile="${2}"
               shift 2
               ;;

           *)
               if [ -n "${2}" ]; then
                   echo "Too many arguments"
                   return 1
               else
                   break
               fi
               ;;
       esac
   done

   if [ -n "${pidfile}" ]; then
      pidlist=`pidofproc -p "${pidfile}" $@`
   else
      pidlist=`pidofproc $@`
   fi

   # Trim trailing blanks
   pidlist=`echo "${pidlist}" | sed -r 's/ +$//'`

   base="${1##*/}"

   if [ -n "${pidlist}" ]; then
      /bin/echo -e "${INFO}${base} is running with Process" \
         "ID(s) ${pidlist}.${NORMAL}"
   else
      if [ -n "${base}" -a -e "/var/run/${base}.pid" ]; then
         /bin/echo -e "${WARNING}${1} is not running but" \
            "/var/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: $@ - Message                                                 #
#                                                                      #
# Return values: Not used                                              #
########################################################################
log_success_msg()
{
    /bin/echo -n -e "${BMPREFIX}${@}"
    /bin/echo -e "${CURS_ZERO}${SUCCESS_PREFIX}${SET_COL}${SUCCESS_SUFFIX}"

    # 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()
{
    /bin/echo -n -e "${BMPREFIX}${@}"
    /bin/echo -e "${CURS_ZERO}${SUCCESS_PREFIX}${SET_COL}${SUCCESS_SUFFIX}"

    echo " OK" >> ${BOOTLOG}

    return 0
}

########################################################################
# log_failure_msg()                                                    #
# Usage: log_failure_msg ["message"]                                   #
#                                                                      #
# Purpose: Print a failure status message to the screen and            #
```

```
#          a boot log file.                                                   #
#                                                                             #
# Inputs: $@ - Message                                                        #
#                                                                             #
# Return values: Not used                                                     #
###############################################################################
log_failure_msg()
{
    /bin/echo -n -e "${BMPREFIX}${@}"
    /bin/echo -e "${CURS_ZERO}${FAILURE_PREFIX}${SET_COL}${FAILURE_SUFFIX}"

    # 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()
{
    /bin/echo -n -e "${BMPREFIX}${@}"
    /bin/echo -e "${CURS_ZERO}${FAILURE_PREFIX}${SET_COL}${FAILURE_SUFFIX}"

    echo "FAIL" >> ${BOOTLOG}

    return 0
}

###############################################################################
# log_warning_msg()                                                           #
# Usage: log_warning_msg ["message"]                                          #
#                                                                             #
# Purpose: Print a warning status message to the screen and                   #
#          a boot log file.                                                   #
#                                                                             #
# Return values: Not used                                                     #
###############################################################################
log_warning_msg()
{
    /bin/echo -n -e "${BMPREFIX}${@}"
    /bin/echo -e "${CURS_ZERO}${WARNING_PREFIX}${SET_COL}${WARNING_SUFFIX}"

    # 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()
{
    /bin/echo -n -e "${BMPREFIX}${@}"
    /bin/echo -e "${CURS_ZERO}${SKIP_PREFIX}${SET_COL}${SKIP_SUFFIX}"
```

```
    # 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()
{
    /bin/echo -n -e "${BMPREFIX}${@}"

    # 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()
{
    /bin/echo -n -e "${@}"

    # 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 failyure as appropriate  #
#                                                                      #
# Purpose: Convenience function to terminate an info message           #
#                                                                      #
# Return values: Not used                                              #
########################################################################
evaluate_retval()
{
   local error_value="${?}"

   if [ ${error_value} = 0 ]; then
      log_success_msg2
   else
      log_failure_msg2
   fi
}
```

```
##############################################################################
# check_signal()                                                             #
# Usage: check_signal [ -{signal} | {signal} ]                               #
#                                                                            #
# Purpose: Check for a valid signal.  This is not defined by any LSB draft,  #
#          however, it is required to check the signals to determine if the  #
#          signals chosen are invalid arguments to the other functions.      #
#                                                                            #
# Inputs: Accepts a single string value in the form or -{signal} or {signal} #
#                                                                            #
# Return values:                                                             #
#       0 - Success (signal is valid                                         #
#       1 - Signal is not valid                                              #
##############################################################################
check_signal()
{
    local valsig

    # Add error handling for invalid signals
    valsig="-ALRM -HUP -INT -KILL -PIPE -POLL -PROF -TERM -USR1 -USR2"
    valsig="${valsig} -VTALRM -STKFLT -PWR -WINCH -CHLD -URG -TSTP -TTIN"
    valsig="${valsig} -TTOU -STOP -CONT -ABRT -FPE -ILL -QUIT -SEGV -TRAP"
    valsig="${valsig} -SYS -EMT -BUS -XCPU -XFSZ -0 -1 -2 -3 -4 -5 -6 -8 -9"
    valsig="${valsig} -11 -13 -14 -15"

    echo "${valsig}" | grep -- " ${1} " > /dev/null

    if [ "${?}" -eq "0" ]; then
        return 0
    else
        return 1
    fi
}

##############################################################################
# check_sig_type()                                                           #
# Usage: check_signal [ -{signal} | {signal} ]                               #
#                                                                            #
# Purpose: Check if signal is a program termination signal or a control signal #
#          This is not defined by any LSB draft, however, it is required to  #
#          check the signals to determine if they are intended to end a      #
#          program or simply to control it.                                  #
#                                                                            #
# Inputs: Accepts a single string value in the form or -{signal} or {signal} #
#                                                                            #
# Return values:                                                             #
#       0 - Signal is used for program termination                           #
#       1 - Signal is used for program control                               #
##############################################################################
check_sig_type()
{
    local valsig

    # The list of termination signals (limited to generally used items)
    valsig="-ALRM -INT -KILL -TERM -PWR -STOP -ABRT -QUIT -2 -3 -6 -9 -14 -15"
```

```
    echo "${valsig}" | grep -- " ${1} " > /dev/null

    if [ "${?}" -eq "0" ]; then
        return 0
    else
        return 1
    fi
}

########################################################################################
# wait_for_user()                                                                      #
#                                                                                      #
# Purpose: Wait for the user to respond if not a headless system                       #
#                                                                                      #
########################################################################################
wait_for_user()
{
   # Wait for the user by default
   [ "${HEADLESS=0}" = "0" ] && read ENTER
   return 0
}

########################################################################################
# is_true()                                                                            #
#                                                                                      #
# Purpose: Utility to test if a variable is true | yes | 1                             #
#                                                                                      #
########################################################################################
is_true()
{
   [ "$1" = "1" ] || [ "$1" = "yes" ] || [ "$1" = "true" ] ||  [ "$1" = "y" ] ||
   [ "$1" = "t" ]
}

# End /lib/lsb/init-functions
```

D.3. /etc/rc.d/init.d/mountvirtfs

```
#!/bin/sh
########################################################################
# Begin mountvirtfs
#
# Description : Mount proc, sysfs, and run
#
# Authors     : Gerard Beekmans - gerard@linuxfromscratch.org
#               DJ Lucas - dj@linuxfromscratch.org
# Update      : Bruce Dubbs - bdubbs@linuxfromscratch.org
#
# Version     : LFS 7.0
#
########################################################################

### BEGIN INIT INFO
# Provides:            mountvirtfs
# Required-Start:
# Should-Start:
```

```
# Required-Stop:
# Should-Stop:
# Default-Start:       S
# Default-Stop:
# Short-Description:   Mounts /sys and /proc virtual (kernel) filesystems.
#                      Mounts /run (tmpfs) and /dev (devtmpfs).
# Description:         Mounts /sys and /proc virtual (kernel) filesystems.
#                      Mounts /run (tmpfs) and /dev (devtmpfs).
# X-LFS-Provided-By:   LFS
### END INIT INFO

. /lib/lsb/init-functions

case "${1}" in
   start)
      # Make sure /run is available before logging any messages
      if ! mountpoint /run >/dev/null; then
         mount /run || failed=1
      fi

      mkdir -p /run/lock /run/shm
      chmod 1777 /run/shm /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

      ln -sfn /run/shm /dev/shm

      (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 sysctl
# Should-Start:
# Required-Stop:
# Should-Stop:
# Default-Start:       S
# Default-Stop:
# Short-Description:   Loads required modules.
# Description:         Loads modules listed in /etc/sysconfig/modules.
# X-LFS-Provided-By:   LFS
### END INIT INFO

# Assure that the kernel has module support.
[ -e /proc/modules ] || exit 0

. /lib/lsb/init-functions

case "${1}" in
   start)
      # Exit if there's no modules file or there are no
      # valid entries
      [ -r /etc/sysconfig/modules ]             || exit 0
      egrep -qv '^($|#)' /etc/sysconfig/modules || exit 0

      log_info_msg "Loading modules:"

      # Only try to load modules if the user has actually given us
      # some modules to load.

      while read module args; do

         # Ignore comments and blank lines.
         case "$module" in
            ""|"#"*) continue ;;
         esac

         # Attempt to load the module, passing any arguments provided.
         modprobe ${module} ${args} >/dev/null
```

```
         # Print the module name if successful, otherwise take note.
         if [ $? -eq 0 ]; then
            log_info_msg2 " ${module}"
         else
            failedmod="${failedmod} ${module}"
         fi
      done < /etc/sysconfig/modules

      # Print a message about successfully loaded modules on the correct line.
      log_success_msg2

      # Print a failure message with a list of any modules that
      # may have failed to load.
      if [ -n "${failedmod}" ]; then
         log_failure_msg "Failed to load modules:${failedmod}"
         exit 1
      fi
      ;;

   *)
      echo "Usage: ${0} {start}"
      exit 1
      ;;
esac

exit 0

# End modules
```

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
#
# Version     : LFS 7.0
#
########################################################################

### BEGIN INIT INFO
# Provides:            udev $time
# Required-Start:
# Should-Start:        modules
# Required-Stop:
# Should-Stop:
# Default-Start:       S
# Default-Stop:
# Short-Description:   Populates /dev with device nodes.
# Description:         Mounts a tempfs on /dev and starts the udevd daemon.
#                      Device nodes are created as defined by udev.
```

```
# X-LFS-Provided-By:   LFS
### END INIT INFO

. /lib/lsb/init-functions

case "${1}" in
   start)
      log_info_msg "Populating /dev with device nodes... "
      if ! grep -q '[[:space:]]sysfs' /proc/mounts; then
         log_failure_msg2
         msg="FAILURE:\n\nUnable to create "
         msg="${msg}devices without a SysFS filesystem\n\n"
         msg="${msg}After you press Enter, this system "
         msg="${msg}will be halted and powered off.\n\n"
         log_info_msg "$msg"
         log_info_msg "Press Enter to continue..."
         wait_for_user
         /etc/rc.d/init.d/halt stop
      fi

      # Start the udev daemon to continually watch for, and act on,
      # uevents
      /sbin/udevd --daemon

      # Now traverse /sys in order to "coldplug" devices that have
      # already been discovered
      /sbin/udevadm trigger --action=add    --type=subsystems
      /sbin/udevadm trigger --action=add    --type=devices
      /sbin/udevadm trigger --action=change --type=devices

      # Now wait for udevd to process the uevents we triggered
      if ! is_true "$OMIT_UDEV_SETTLE"; then
         /sbin/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:
# Default-Start:       S
# Default-Stop:        0 6
# Short-Description:   Mounts and unmounts swap partitions.
# Description:         Mounts and unmounts swap partitions defined in
#                      /etc/fstab.
# X-LFS-Provided-By:   LFS
### END INIT INFO

. /lib/lsb/init-functions

case "${1}" in
   start)
      log_info_msg "Activating all swap files/partitions..."
      swapon -a
      evaluate_retval
      ;;

   stop)
      log_info_msg "Deactivating all swap files/partitions..."
      swapoff -a
      evaluate_retval
      ;;

   restart)
      ${0} stop
      sleep 1
      ${0} start
      ;;

   status)
      log_success_msg "Retrieving swap status."
      swapon -s
      ;;

   *)
      echo "Usage: ${0} {start|stop|restart|status}"
      exit 1
      ;;
esac
```

```
exit 0

# End swap
```

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 BLFS
### END INIT INFO

. /lib/lsb/init-functions

[ -r /etc/sysconfig/clock ] && . /etc/sysconfig/clock


case "${UTC}" in
   yes|true|1)
      CLOCKPARAMS="${CLOCKPARAMS} --utc"
      ;;

   no|false|0)
      CLOCKPARAMS="${CLOCKPARAMS} --localtime"
      ;;

esac

case ${1} in
   start)
      hwclock --hctosys ${CLOCKPARAMS} >/dev/null
      ;;
```

```
   stop)
      log_info_msg "Setting hardware clock..."
      hwclock --systohc ${CLOCKPARAMS} >/dev/null
      evaluate_retval
      ;;

   *)
      echo "Usage: ${0} {start|stop}"
      exit 1
      ;;

esac

exit 0
```

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 $time
# Should-Start:
# Required-Stop:
# Should-Stop:
# Default-Start:       S
# Default-Stop:
# Short-Description:   Checks local filesystems before mounting.
# Description:         Checks local filesystmes before mounting.
# X-LFS-Provided-By:   LFS
### END INIT INFO
```

```
. /lib/lsb/init-functions

case "${1}" in
   start)
      if [ -f /fastboot ]; then
         msg="/fastboot found, will omit "
         msg="${msg} file system checks as requested.\n"
         log_info_msg "${msg}"
         exit 0
      fi

      log_info_msg "Mounting root file system in read-only mode... "
      mount -n -o remount,ro / >/dev/null

      if [ ${?} != 0 ]; then
         log_failure_msg2
         msg="\n\nCannot check root "
         msg="${msg}filesystem because it could not be mounted "
         msg="${msg}in read-only mode.\n\n"
         msg="${msg}After you press Enter, this system will be "
         msg="${msg}halted and powered off.\n\n"
         log_failure_msg "${msg}"

         log_info_msg "Press Enter to continue..."
         wait_for_user
         /etc/rc.d/init.d/halt stop
      else
         log_success_msg2
      fi

      if [ -f /forcefsck ]; then
         msg="/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="${msg}were found and have been been "
         msg="${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 stop
      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:
# Required-Stop:       swap
# Should-Stop:
# Default-Start:       S
# Default-Stop:        0 6
# Short-Description:   Mounts/unmounts local filesystems defined in /etc/fstab.
# Description:         Remounts root filesystem read/write and mounts all
#                      remaining local filesystems defined in /etc/fstab on
#                      start.  Remounts root filesystem read-only and unmounts
#                      remaining filesystems on stop.
# X-LFS-Provided-By:   LFS
### END INIT INFO

. /lib/lsb/init-functions

case "${1}" in
   start)
      log_info_msg "Remounting root file system in read-write mode..."
      mount --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..."
      mount --all --test-opts no_netdev >/dev/null
      evaluate_retval
      exit $failed
      ;;

   stop)
```

```
      # Don't unmount virtual file systems like /run
      log_info_msg "Unmounting all other currently mounted file systems..."
      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
      ;;

   *)
      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
# Required-Stop:
# Should-Stop:
# Default-Start:       S
# Default-Stop:
# Short-Description:   Replays failed uevents and creates additional devices.
# Description:         Replays any failed uevents that were skipped due to
#                      slow hardware initialization, and creates those needed
#                      device nodes
# X-LFS-Provided-By:   LFS
### END INIT INFO

. /lib/lsb/init-functions
```

```
case "${1}" in
   start)
      log_info_msg "Retrying failed uevents, if any..."

      # As of udev-186, the --run option is no longer valid
      #rundir=$(/sbin/udevadm info --run)
      rundir=/run/udev
      # From Debian: "copy the rules generated before / was mounted
      # read-write":

      for file in ${rundir}/tmp-rules--*; do
         dest=${file##*tmp-rules--}
         [ "$dest" = '*' ] && break
         cat $file >> /etc/udev/rules.d/$dest
         rm -f $file
      done

      # Re-trigger the uevents that may have failed,
      # in hope they will succeed now
      /bin/sed -e 's/#.*$//' /etc/sysconfig/udev_retry | /bin/grep -v '^$' | \
      while read line ; do
         for subsystem in $line ; do
            /sbin/udevadm trigger --subsystem-match=$subsystem --action=add
         done
      done

      # Now wait for udevd to process the uevents we triggered
      if ! is_true "$OMIT_UDEV_RETRY_SETTLE"; then
         /sbin/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 /var/run, /var/lock, and
#                      optionally, /tmp.  cleanfs also creates /var/run/utmp
#                      and any files defined in /etc/sysconfig/createfiles.
# X-LFS-Provided-By:   LFS
### END INIT INFO

. /lib/lsb/init-functions

# Function to create files/directory on boot.
create_files()
{
   # Input to file descriptor 9 and output to stdin (redirection)
   exec 9>&0 < /etc/sysconfig/createfiles

   while read name type perm usr grp dtype maj min junk
   do
      # Ignore comments and blank lines.
      case "${name}" in
         ""|\#*) continue ;;
      esac

      # Ignore existing files.
      if [ ! -e "${name}" ]; then
         # Create stuff based on its type.
         case "${type}" in
            dir)
               mkdir "${name}"
               ;;
            file)
               :> "${name}"
               ;;
            dev)
               case "${dtype}" in
                  char)
                     mknod "${name}" c ${maj} ${min}
                     ;;
                  block)
                     mknod "${name}" b ${maj} ${min}
                     ;;
                  pipe)
                     mknod "${name}" p
                     ;;
                  *)
                     log_warning_msg "\nUnknown device type: ${dtype}"
```

```
                      ;;
               esac
               ;;
            *)
               log_warning_msg "\nUnknown type: ${type}"
               continue
               ;;
         esac

         # Set up the permissions, too.
         chown ${usr}:${grp} "${name}"
         chmod ${perm} "${name}"
      fi
   done

   # Close file descriptor 9 (end redirection)
   exec 0>&9 9>&-
   return 0
}

case "${1}" in
   start)
      log_info_msg "Cleaning file systems:"

      if [ "${SKIPTMPCLEAN}" = "" ]; then
         log_info_msg2 " /tmp"
         cd /tmp &&
         find . -xdev -mindepth 1 ! -name lost+found -delete || failed=1
      fi

      > /var/run/utmp

      if grep -q '^utmp:' /etc/group ; then
         chmod 664 /var/run/utmp
         chgrp utmp /var/run/utmp
      fi

      (exit ${failed})
      evaluate_retval

      if egrep -qv '^(#|$)' /etc/sysconfig/createfiles 2>/dev/null; then
         log_info_msg "Creating files and directories... "
         create_files      # Always returns 0
         evaluate_retval
      fi

      exit $failed
      ;;
   *)
      echo "Usage: ${0} {start}"
      exit 1
      ;;
esac

# End cleanfs
```

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:
# Should-Start:        $local_fs
# Required-Stop:
# Should-Stop:
# Default-Start:       S
# Default-Stop:
# Short-Description:   Sets up a localised console.
# Description:         Sets up fonts and language settings for the user's
#                      local as defined by /etc/sysconfig/console.
# X-LFS-Provided-By:   LFS
### END INIT INFO

. /lib/lsb/init-functions

# Native English speakers probably don't have /etc/sysconfig/console at all
[ -r /etc/sysconfig/console ] && . /etc/sysconfig/console

is_true()
{
   [ "$1" = "1" ] || [ "$1" = "yes" ] || [ "$1" = "true" ]
}

failed=0

case "${1}" in
   start)
      # See if we need to do anything
      if [ -z "${KEYMAP}"         ] && [ -z "${KEYMAP_CORRECTIONS}" ] &&
         [ -z "${FONT}"           ] && [ -z "${LEGACY_CHARSET}"     ] &&
         ! is_true "${UNICODE}"; then
         exit 0
      fi

      # There should be no bogus failures below this line!
      log_info_msg "Setting up Linux console..."

      # Figure out if a framebuffer console is used
```

```
      [ -d /sys/class/graphics/fb0 ] && use_fb=1 || use_fb=0

      # Figure out the command to set the console into the
      # desired mode
      is_true "${UNICODE}" &&
         MODE_COMMAND="echo -en '\033%G' && kbd_mode -u" ||
         MODE_COMMAND="echo -en '\033%@\033(K' && kbd_mode -a"

      # On framebuffer consoles, font has to be set for each vt in
      # UTF-8 mode. This doesn't hurt in non-UTF-8 mode also.

      ! is_true "${use_fb}" || [ -z "${FONT}" ] ||
         MODE_COMMAND="${MODE_COMMAND} && setfont ${FONT}"

      # Apply that command to all consoles mentioned in
      # /etc/inittab. Important: in the UTF-8 mode this should
      # happen before setfont, otherwise a kernel bug will
      # show up and the unicode map of the font will not be
      # used.

      for TTY in `grep '^[^#].*respawn:/sbin/agetty' /etc/inittab |
         grep -o '\btty[[:digit:]]*\b'`
      do
         openvt -f -w -c ${TTY#tty} -- \
            /bin/sh -c "${MODE_COMMAND}" || failed=1
      done

      # Set the font (if not already set above) and the keymap
      [ "${use_fb}" == "1" ] || [ -z "${FONT}" ] || setfont $FONT || failed=1

      [ -z "${KEYMAP}" ] ||
         loadkeys ${KEYMAP} >/dev/null 2>&1 ||
         failed=1

      [ -z "${KEYMAP_CORRECTIONS}" ] ||
         loadkeys ${KEYMAP_CORRECTIONS} >/dev/null 2>&1 ||
         failed=1

      # Convert the keymap from $LEGACY_CHARSET to UTF-8
      [ -z "$LEGACY_CHARSET" ] ||
         dumpkeys -c "$LEGACY_CHARSET" | loadkeys -u >/dev/null 2>&1 ||
         failed=1

      # If any of the commands above failed, the trap at the
      # top would set $failed to 1
      ( exit $failed )
      evaluate_retval

      exit $failed
      ;;

   *)
      echo "Usage:  ${0} {start}"
      exit 1
      ;;
esac
```

```
# End console
```

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:      $local_fs
# Should-Start:
# Required-Stop:
# Should-Stop:
# Default-Start:       S
# Default-Stop:        0 6
# Short-Description:   Starts the local network.
# Description:         Sets the hostname of the machine and starts the
#                      loopback interface.
# X-LFS-Provided-By:   LFS
### END INIT INFO

. /lib/lsb/init-functions
[ -r /etc/sysconfig/network ] && . /etc/sysconfig/network
[ -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:
# 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
#
# Version     : LFS 7.0
#
########################################################################

### BEGIN INIT INFO
# Provides:            $syslog
# Required-Start:      localnet
# Should-Start:
# Required-Stop:       $local_fs sendsignals
# Should-Stop:
# Default-Start:       3 4 5
# Default-Stop:        0 1 2 6
# Short-Description:   Starts kernel and system log daemons.
# Description:         Starts kernel and system log daemons.
#                      /etc/fstab.
# X-LFS-Provided-By:   LFS
### END INIT INFO

# Note: sysklogd is not started in runlevel 2 due to possible
# remote logging configurations

. /lib/lsb/init-functions

case "${1}" in
```

```
   start)
      log_info_msg "Starting system log daemon..."
      parms=${SYSKLOGD_PARMS-'-m 0'}
      start_daemon /sbin/syslogd $parms
      evaluate_retval

      log_info_msg "Starting kernel log daemon..."
      start_daemon /sbin/klogd
      evaluate_retval
      ;;

   stop)
      log_info_msg "Stopping kernel log daemon..."
      killproc /sbin/klogd
      evaluate_retval

      log_info_msg "Stopping system log daemon..."
      killproc /sbin/syslogd
      evaluate_retval
      ;;

   reload)
      log_info_msg "Reloading system log daemon config file..."
      pid=`pidofproc syslogd`
      kill -HUP "${pid}"
      evaluate_retval
      ;;

   restart)
      ${0} stop
      sleep 1
      ${0} start
      ;;

   status)
      statusproc /sbin/syslogd
      statusproc klogd
      ;;

   *)
      echo "Usage: ${0} {start|stop|reload|restart|status}"
      exit 1
      ;;
esac

exit 0

# End sysklogd
```

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 swap localnet
# Should-Start:        $syslog
# Required-Stop:       $local_fs swap localnet
# Should-Stop:         $syslog
# Default-Start:       3 4 5
# Default-Stop:        0 1 2 6
# Short-Description:   Starts and configures network interfaces.
# Description:         Starts and configures network interfaces.
# X-LFS-Provided-By:   LFS
### END INIT INFO

case "${1}" in
   start)
      # Start all network interfaces
      for file in /etc/sysconfig/ifconfig.*
      do
         interface=${file##*/ifconfig.}

         # Skip if $file is * (because nothing was found)
         if [ "${interface}" = "*" ]
         then
            continue
         fi

         /sbin/ifup ${interface}
      done
      ;;

   stop)
      # 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

         /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 "${1}" in
   stop)
      log_info_msg "Sending all processes the TERM signal..."
      killall5 -15
      error_value=${?}

      sleep ${KILLDELAY}

      if [ "${error_value}" = 0 -o "${error_value}" = 2 ]; then
         log_success_msg
      else
         log_failure_msg
      fi

      log_info_msg "Sending all processes the KILL signal..."
      killall5 -9
      error_value=${?}

      sleep ${KILLDELAY}

      if [ "${error_value}" = 0 -o "${error_value}" = 2 ]; then
         log_success_msg
      else
         log_failure_msg
      fi
      ;;

   *)
      echo "Usage: ${0} {stop}"
      exit 1
      ;;

esac

exit 0

# End sendsignals
```

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
# Update      : Bruce Dubbs - bdubbs@linuxfromscratch.org
#
# Version     : LFS 7.0
#
########################################################################

### BEGIN INIT INFO
```

```
# Provides:            reboot
# Required-Start:
# Should-Start:
# Required-Stop:
# Should-Stop:
# Default-Start:       6
# Default-Stop:
# Short-Description:   Reboots the system.
# Description:         Reboots the System.
# X-LFS-Provided-By:   LFS
### END INIT INFO

. /lib/lsb/init-functions

case "${1}" in
   stop)
      log_info_msg "Restarting system..."
      reboot -d -f -i
      ;;

   *)
      echo "Usage: ${0} {stop}"
      exit 1
      ;;

esac

# End reboot
```

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
#
# Version     : LFS 7.0
#
########################################################################

### BEGIN INIT INFO
# Provides:            halt
# Required-Start:
# Should-Start:
# Required-Stop:
# Should-Stop:
# Default-Start:       0
# Default-Stop:
# Short-Description:   Halts the system.
# Description:         Halts the System.
# X-LFS-Provided-By:   LFS
```

```
### END INIT INFO

case "${1}" in
   stop)
      halt -d -f -i -p
      ;;

   *)
      echo "Usage: {stop}"
      exit 1
      ;;
esac

# End halt
```

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..."
      start_daemon fully_qualified_path
      ;;

   stop)
      log_info_msg "Stopping..."
      killproc fully_qualified_path
      ;;
```

```
   restart)
      ${0} stop
      sleep 1
      ${0} start
      ;;

   *)
      echo "Usage: ${0} {start|stop|restart}"
      exit 1
      ;;
esac

exit 0

# End scriptname
```

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 deivce
#               pipe creates a pipe, this will ignore the <major> and
#           <minor> fields
#         <major> and <minor> are the major and minor numbers used for
#     the device.
#########################################################################

# End /etc/sysconfig/createfiles
```

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
```

Appendix E. Правила конфигурации Udev

Правила в этом приложении перечислены для удобства. Установка обычно выполняются с помощью инструкций в Section 6.77, “Eudev-3.2.8”.

E.1. 55-lfs.rules

```
# /etc/udev/rules.d/55-lfs.rules: Rule definitions for LFS.

# Core kernel devices

# This causes the system clock to be set as soon as /dev/rtc becomes available.
SUBSYSTEM=="rtc", ACTION=="add", MODE="0644", RUN+="/etc/rc.d/init.d/setclock start"
KERNEL=="rtc", ACTION=="add", MODE="0644", RUN+="/etc/rc.d/init.d/setclock start"

# Comms devices

KERNEL=="ippp[0-9]*",       GROUP="dialout"
KERNEL=="isdn[0-9]*",       GROUP="dialout"
KERNEL=="isdnctrl[0-9]*",   GROUP="dialout"
KERNEL=="dcbri[0-9]*",      GROUP="dialout"
```

Appendix F. Лицензия LFS

F.1. С указанием авторства-С сохранением условий версии 4.0 Международная (Creative Commons License)

Использование Публичных Лицензий Creative Commons

Публичная лицензия Creative Commons с указанием авторства-С сохранением условий версии 4.0 Международная

Important

Корпорация Creative Commons ("Creative Commons") не является юридической фирмой, не оказывает юридических услуг или консультаций. Распространение публичных лицензий Creative Commons не порождает отношений, аналогичных отношениям между юристом и клиентом, или каких-либо иных отношений. Creative Commons предоставляет доступ к своим лицензиям и всей сопутствующей информации на основе принципа "как есть". Creative Commons не предоставляет каких-либо гарантий в отношении лицензий, любого материала, предоставляемого на условиях таких лицензий, или любой сопутствующей информации. Creative Commons, насколько это возможно, не будет нести никакой ответственности за ущерб, возникший в результате их применения.

Лицензия

Публичные лицензии Creative Commons представляют собой стандартный набор условий, которые создатели и иные правообладатели могут использовать для предоставления оригинальных авторских произведений и других материалов, которые охраняются авторским правом и некоторыми другими правами, указанными в тексте нижеизложенной публичной лицензии. Приведенные ниже соображения следует рассматривать исключительно в информационных целях; они не являются исчерпывающими, а также не являются составной частью наших лицензий.

Информация для лицензиаров

Наши публичные лицензии предназначены для использования лицами, которые вправе предоставлять публичное разрешение на использование материалов способами, использование которых в противном случае было бы ограничено авторским правом и некоторыми другими правами. Наши лицензии являются безотзывными. Лицензиары должны прочитать и понять условия лицензии, которую они выбирают, до начала ее применения. Лицензиары должны обеспечить наличие всех необходимых прав до начала применения наших лицензий таким образом, чтобы неограниченный круг лиц мог пользоваться материалами в соответствии с ожиданиями. Лицензиары должны четко обозначить любой материал, который не предоставляется на условиях такой лицензии, включая любой иной материал, самостоятельно предоставляемый на условиях лицензий CC, или материал, используемый на основе исключений из авторского права или в силу ограничения авторского права.

Информация для неограниченного круга лиц

Используя одну из наших публичных лицензий, лицензиар предоставляет неограниченному кругу лиц разрешение использовать лицензируемый материал на определенных условиях. Если разрешение лицензиара по какой-либо причине не требуется, например, при наличии любых применимых исключений из авторского права или ограничений такого права, такое использование не регулируется публичной лицензией. Наши лицензии предоставляют только разрешения в отношении авторских прав и некоторыми других аналогичных прав, при наличии права предоставлять такие разрешения у Лицензиара. Использование лицензируемого материала может быть ограничено по другим причинам, в том числе, если авторские или иные права в отношении материала принадлежат другим лицам. Лицензиар может устанавливать специальные пожелания, например, просить, чтобы все изменения, вносимые в лицензируемый материал, были отмечены или описаны. Хотя это и не требуется нашими лицензиями, рекомендуем выполнять соответствующие пожелания в разумных пределах. Более подробная информация для неограниченного круга лиц.

1. Раздел 1 – Определения

   a. Адаптированный Материал означает материал, охраняемый Авторским Правом И Другими Схожими Правами, производный от Лицензируемого Материала или основанный на Лицензируемом Материале, который содержит перевод, измененный вариант, аранжировку, преобразованный вариант или иную переработку Лицензируемого Материала, таким образом, который требует разрешения Лицензиара в соответствии с Авторским Правом И Другими Схожими Правами. Для целей настоящей Публичной Лицензии, в том случае, если Лицензируемый Материал является музыкальным произведением, исполнением или фонограммой, синхронизация такого Лицензируемого Материала с движущимся изображением всегда порождает "Адаптированный Материал".

   b. "Лицензия на Адаптированный Материал означает лицензию, которую Вы применяете в отношении Авторского Права И Других Схожих Прав на Ваш вклад в создание Адаптированного Материала в соответствии с условиями настоящей Публичной Лицензии.

   c. Лицензия, Совместимая С Лицензией Creative Commons BY-SA означает лицензию, входящую в перечень лицензий, размещенный по адресу: creativecommons.org/compatiblelicenses, которая признается Creative Commons по существу эквивалентом настоящей Публичной Лицензии.

   d. Авторское Право И Другие Схожие Права означают авторские и/или другие аналогичные права, тесно связанные с авторскими правами, включая, но не ограничиваясь этим, права на исполнение, сообщение передач организаций вещания, фонограмму и Права Sui Generis на Базы Данных независимо от того, каким образом права обозначаются или классифицируются. Для целей настоящей Публичной Лицензии права, указанные в подпунктах 1 и 2 пункта (b) Раздела 2 не являются Авторским Правом И Другими Схожими Правами.

   e. Эффективные Технические Меры означают меры, которые при отсутствии надлежащих полномочий нельзя обойти в соответствии с законодательством, направленным на выполнение обязательств в соответствии со Статьей 11 Договора Всемирной организации интеллектуальной собственности (ВОИС) по авторскому праву, принятого 20 декабря 1996 года и/или иными подобными международными соглашениями.

f. Исключения И Ограничения означают свободное использование (fair use, fair dealing) и/или любые иные исключения из Авторского Права И Других Схожих Прав или ограничения таких прав, которые применяются в отношении Вашего использования Лицензируемого Материала.

g. Элементы Лицензии означают атрибуты лицензии, перечисленные в названии настоящей Публичной Лицензии Creative Commons. Элементами Лицензии настоящей Публичной Лицензии являются: С указанием авторства и С сохранением условий.

h. Лицензируемый Материал означает произведение искусства или литературы, базу данных или другой материал, в отношении которого Лицензиар применил настоящую Публичную Лицензию.

i. Лицензируемые Права означают права, предоставляемые Вам в соответствии с условиями настоящей Публичной Лицензии в объеме, ограниченном всеми Авторскими Правами И Другими Схожими Правами, которые применимы к Вашему использованию Лицензируемого Материала и которые Лицензиар вправе Вам предоставить.

j. Лицензиар означает физическое лицо (физические лица) или юридическое лицо (юридические лица), предоставляющее права на условиях настоящей Публичной Лицензии

k. Предоставление означает предоставление материала неограниченному кругу лиц любыми средствами или способами, для использования которых требуется разрешение согласно Лицензируемых Прав, в том числе воспроизведение, публичный показ, публичное исполнение, распространение, сообщение или импорт, а также доведение материала до всеобщего сведения таким образом, при котором любое лицо может иметь доступ к нему из любого места и в любое время по собственному выбору.

l. Права Sui Generis на Базы Данных означают права, не являющиеся авторским правом, проистекающие из Директивы 96/9/CE Европейского парламента и Совета от 11 марта 1996 года о правовой охране баз данных, с учетом изменений и/или исправлений, а также и другие схожие по существу права где-либо в мире.

m. Вы означает физическое или юридическое лицо, осуществляющее Лицензируемые Права в соответствии с настоящей Публичной Лицензией. Термины "Ваш", "Ваши", "Вам", "Вами" имеют соответствующее значение.

2. Раздел 2 – Объем лицензии.

a. Предоставление лицензии.

В соответствии с условиями настоящей Публичной Лицензии Лицензиар предоставляет Вам действующую на территории всех стран мира, безвозмездную, без права сублицензирования, неисключительную, не подлежащую отмене лицензию на осуществление Лицензируемых Прав на Лицензируемый Материал путем: воспроизведения и Предоставления Лицензируемого Материала целиком или в части; а также создания, воспроизведения и Предоставления Адаптированного Материала.

Исключения И Ограничения. Во избежание неоднозначного толкования, если Исключения и Ограничения применяются в отношении Вашего способа использования Лицензируемого материала, настоящая Публичная Лицензия не применяется, и Вы не обязаны выполнять ее условия.

Срок действия лицензии. Настоящая Публичная Лицензия действует в течение срока, указанного в пункте (a) Раздела 6.

Носители и форматы: разрешение на внесение технических изменений. Лицензиар предоставляет Вам право осуществлять Лицензируемые Права с использованием всех известных носителей и форматов, а также носителей и форматов, которые будут созданы в будущем, и вносить с этой целью любые необходимые технические изменения. Лицензиар отказывается от и/или соглашается не осуществлять какие-либо права или полномочия, позволяющие запретить внесение Вами технических изменений, необходимых для осуществления Лицензируемых Прав, включая технические изменения, необходимые для обхода Эффективных Технических Мер. Для целей настоящей Публичной Лицензии внесение изменений, разрешенных в подпункте (4) пункта (a) Раздела 2, само по себе не является созданием Адаптированного Материала.

Последующие получатели.Оферта от Лицензиара – Лицензируемый Материал. Каждый получатель Лицензируемого Материала автоматически получает оферту от Лицензиара на осуществление Лицензируемых Прав в соответствии с условиями настоящей Публичной Лицензии.Дополнительная оферта от Лицензиара – Адаптированный Материал. Каждый получатель Адаптированного Материала от Вас автоматически получает оферту от Лицензиара на осуществление Лицензируемых Прав на Адаптированный Материал в соответствии с условиями Лицензии на Адаптированный Материал, применяемой Вами.Отсутствие ограничений на последующее использование. Вы не можете предлагать или устанавливать какие-либо дополнительные или иные условия, или применять Эффективные Технические Меры в отношении Лицензируемого Материала, если это ограничивает осуществление Лицензируемых Прав любым получателем Лицензируемого Материала.

Отсутствие одобрения. Никакое положение настоящей Публичной Лицензии не является разрешением и не может быть истолковано как разрешение утверждать или предполагать, что Вы или использование Вами Лицензируемого Материала каким-либо образом связаны с Лицензиаром, имеете финансовую поддержку, одобрение или официальный статус, предоставленные Лицензиаром или иными лицами, указанными Лицензиаром для указания авторства, как предусмотрено в подпункте (i) (A) (1) пункта (a) Раздела 3.

b. Иные права.

Личные неимущественные права, такие как право на неприкосновенность произведения, а также права на публичность и изображение гражданина, неприкосновенность частной жизни или иные аналогичные личные права не предоставляются на основе настоящей Публичной Лицензии. Тем не менее, в максимально возможной степени Лицензиар отказывается или соглашается не осуществлять любые такие права, принадлежащие ему, в объеме, необходимом, чтобы позволить Вам осуществлять Лицензируемые Права, но не иначе.

Патентные права и права на товарные знаки и знаки обслуживания не предоставляются по настоящей Публичной Лицензии.

В той мере, в которой это возможно, Лицензиар отказывается от любого права на получение от Вас вознаграждения за осуществление Вами Лицензируемых Прав, как непосредственно, так и через любые организации по коллективному управлению правами

или любую добровольную, обязательную государственную или принудительную систему лицензирования. Во всех иных случаях Лицензиар сохраняет право на получение такого вознаграждения.

Раздел 3 – Условия лицензии.

1. Вы можете осуществлять Лицензируемые Права исключительно при условии соблюдения следующих условий.

   Указание авторства. Если Вы Предоставляете Лицензируемый Материал (в том числе в измененном виде), Вы должны: сохранить следующие сведения, если они предоставлены Лицензиаром вместе с Лицензируемым Материалом: информацию об создателе (создателях) Лицензируемого Материала, а также любых других лицах, указанных Лицензиаром, обладающих правом на указание авторства, любым разумным способом, по требованию Лицензиара (в том числе с использованием псевдонима, если таковой указан);уведомление об авторском праве;уведомление об использовании настоящей Публичной Лицензии;уведомление об отказе от гарантий;Унифицированный Идентификатор Ресурса (URI) или гиперссылку на Лицензируемый Материал, в той мере, в которой это практически выполнимо;указать, если Вами внесены изменения в Лицензируемый Материал, и сохранить указание на любые предыдущие изменения; а также указать, что Лицензируемый Материал предоставляется на условиях настоящей Публичной Лицензии, и предоставить текст, или Унифицированный Идентификатор Ресурса (URI), или гиперссылку на настоящую Публичную Лицензию. Вы можете выполнить положения подпункта (1) пункта (a) Раздела 3 любым разумным способом в зависимости от носителя, способа и контекста, посредством которых вы Предоставляете Лицензируемый Материал. Например, разумным признается выполнение данного условия путем указания Унифицированного Идентификатора Ресурса (URI) или гиперссылки на ресурс, который содержит необходимую информацию. По требованию Лицензиара Вы должны, насколько это практически выполнимо, удалить любую информацию, указанную в подпункте (A) (1) пункта (a) Раздела 3.

   С сохранением условий.

   В дополнение к условиям, указанным в пункте (a) Раздела 3, если вы Предоставляете Адаптированный Материал, созданный Вами, то применяются также следующие условия. В качестве Лицензии на Адаптированный Материал, которую Вы применяете, должна быть применена лицензия Creative Commons с теми же Элементами Лицензии, данной версии или более поздней версии, или Лицензия, Совместимая С Лицензией BY-SA. Вы должны включить текст, Унифицированный Идентификатор Ресурса (URI) или гиперссылку на Лицензию на Адаптированный Материал, которую Вы применяете. Вы можете выполнить данное условие любым разумным способом в зависимости от носителя, способа и контекста, посредством которых Вы Предоставляете Адаптированный Материал. Вы не можете предлагать или устанавливать какие-либо дополнительные или иные условия или применять Эффективные Технические Меры в отношении Адаптированного Материала, которые ограничивают осуществление прав, предоставленных в соответствии с Лицензией на Адаптированный Материал, которую Вы применяете.

Раздел 4 – Права Sui Generis на Базы Данных.

1. Если Лицензируемые Права включают Права Sui Generis на Базы Данных, которые применимы в отношении Вашего использования Лицензируемого Материала:

во избежание неоднозначного толкования в соответствии с подпунктом (1) пункта (a) Раздела 2 Вам предоставляются права на извлечение, повторное использование, воспроизведение и Предоставление содержимого базы данных полностью или в существенной части;

если Вы включаете содержимое базы данных полностью или его существенную часть в базу данных, в отношении которой у Вас имеются Права на содержимое баз данных, в этом случае база данных, в отношении которой у Вас имеются Права на содержимое баз данных (но не ее отдельные составляющие), является Адаптированным Материалом, в том числе в целях, указанных в пункте (b) Раздела 3; а также

Вы должны соблюдать условия, изложенные в пункте (a) Раздела 3, если Вы Предоставляете содержимое базы данных полностью или его существенную часть.

Во избежание неоднозначного толкования данный Раздел 4 дополняет и не заменяет Ваши обязательства в соответствии с настоящей Публичной Лицензией, если Лицензируемые Права включают другие Авторские Права И Другие Схожие Права.

Раздел 5 – Отказ от гарантий и ограничение ответственности.

1. Если иное отдельно не оговорено Лицензиаром, насколько это возможно, Лицензиар предлагает Лицензируемый Материал по принципу "как есть" и в том виде, в котором такой материал существует, и не дает никаких заверений или гарантий в отношении Лицензируемого Материала, выраженных в явном виде, предполагаемых, установленных законом или иных, включая, без ограничений, гарантии правового титула, товарной пригодности, пригодности для какой-либо определенной цели, гарантии не нарушения прав, отсутствия скрытых или других дефектов, точности, наличия или отсутствия ошибок, как известных, так и неизвестных, или как поддающихся, так и не поддающихся обнаружению. В том случае, если отказ от гарантий не разрешен полностью или частично, такой отказ может не применяться в отношении Вас.

   В той мере, в которой это возможно, Лицензиар не несет перед Вами никакой ответственности на основании любой правовой доктрины (в том числе, но не ограничиваясь, в результате неосторожности), за какие-либо прямые, особые, непрямые, случайные, косвенные или иные убытки и штрафные выплаты, издержки, затраты или ущерб, возникшие в результате применения настоящей Публичной Лицензии или использования Лицензируемого Материала, даже если Лицензиар был уведомлен о возможности возникновения таких затрат, издержек или убытков. В том случае если ограничение ответственности полностью или частично не допускается, настоящее ограничение может не применяться в отношении Вас.

   Отказ от гарантий и ограничение ответственности, изложенные выше, должны толковаться таким образом, чтобы в максимально допустимой степени соответствовать абсолютному отказу от гарантий и отказу от какой-либо ответственности.

Раздел 6 – Срок действия и прекращение действия.

1. Настоящая Публичная Лицензия действует в течение срока действия Авторского Права И Других Схожих Прав, предоставляемых в соответствии с настоящей Публичной Лицензией. При этом если Вы не соблюдаете какое-либо условие настоящей Публичной Лицензии, действие Ваших прав в соответствии с настоящей Публичной Лицензией автоматически прекращается.

Если Ваше право на использование Лицензируемого Материала прекратилось в соответствии с пунктом (a) Раздела 6, оно считается восстановленным: автоматически в момент устранения Вами нарушения, если такое нарушение устранено в течение 30 дней с момента его обнаружения; или в случае четко выраженного решения Лицензиара о восстановлении права. Во избежание неоднозначного толкования пункт (b) Раздела 6 не затрагивает каких-либо прав Лицензиара на поиск средств правовой защиты от допущенных Вами нарушений условий настоящей Публичной Лицензии.

Во избежание неоднозначного толкования Лицензиар может также предлагать Лицензируемый Материал на иных лицензионных условиях или остановить распространение Лицензируемого Материала в любое время; однако, это не прекращает действия Публичной Лицензии.

Разделы 1, 5, 6, 7 и 8 продолжают действовать после прекращения действия настоящей Публичной Лицензии.

Раздел 7 – Прочие условия.

1. Лицензиар не должен быть связан никакими дополнительными или иными условиями, сообщенными Вами, без его согласия, выраженного в явном виде.

   Любые дополнительные договоренности или понимания, касающиеся Лицензируемого Материала, которые не указаны в настоящей Публичной Лицензии, являются отдельными и независимыми от условий настоящей Публичной Лицензии.

Раздел 8 – Толкование.

1. Во избежание неоднозначного толкования настоящая Публичная Лицензия не может и не должна толковаться как сокращение, ограничение или наложение условий в отношении любого использования Лицензируемого Материала, которое может быть осуществлено на законных основаниях без разрешения, предоставляемого в соответствии с настоящей Публичной Лицензией.

   Если какое-либо положение настоящей Публичной Лицензии считается не имеющим законной силы, оно должно быть, насколько это возможно, автоматически исправлено в минимально необходимой степени для наделения такого положения законной силой. Если такое положение невозможно исправить, оно должно быть исключено из текста настоящей Публичной Лицензии без ущерба для законной силы остальных положений лицензии.

   Никакое условие или положение настоящей Публичной Лицензии не будет считаться отмененным, а нарушение - согласованным, если Лицензиар в явной форме не выразит свое согласие с такой отменой или нарушением.

   Никакое условие или положение настоящей Публичной Лицензии не является и не может быть истолковано как ограничение или отказ от каких-либо привилегий и иммунитетов, применимых в отношении Лицензиара и/или в отношении Вас, включая отказ от судебных процессов в какой-либо юрисдикции или какой-либо подведомственности.

Important

Creative Commons не является стороной её публичных лицензий. Тем не менее, Creative Commons может принять решение об использовании одной из её публичных лицензий в отношении публикуемых Creative Commons материалов и в этом случае будет являться "Лицензиаром". Текст публичных лицензий Creative Commons передан в общественное достояние в соответствии с Заявлением о передаче в общественное достояние CC0. За исключением ограниченного количества случаев, когда материал предоставляется на условиях публичной лицензии Creative Commons или в иных случаях, предусмотренных политикой Creative Commons, опубликованной по адресу: creativecommons.org/policies, Creative Commons не разрешает использовать товарный знак / знак обслуживания "Creative Commons" или любой другой товарный знак / знак обслуживания или логотип Creative Commons без своего предварительного письменного согласия, включая, но не ограничиваясь, в связи с любыми неразрешенными изменениями, внесенные в какие-либо из её публичных лицензий или любые другие договоренности, соглашения или договоры, касающиеся использования лицензируемого материала. Во избежание неоднозначного толкования данный параграф не является частью публичных лицензий.

Вы можете обратиться в корпорацию Creative Commons по адресу: *https://creativecommons.org/*.

F.2. Лицензия MIT

Данная лицензия разрешает, безвозмездно, лицам, получившим копию данного программного обеспечения и сопутствующей документации (в дальнейшем именуемыми "Программное Обеспечение"), использовать Программное Обеспечение без ограничений, включая неограниченное право на использование, копирование, изменение, объединение, публикацию, распространение, сублицензирование и/или продажу копий Программного Обеспечения, также как и лицам, которым предоставляется данное Программное Обеспечение, при соблюдении следующих условий:

Вышеупомянутый копирайт и данные условия должны быть включены во все копии или значимые части данного Программного Обеспечения.

ДАННОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ПРЕДОСТАВЛЯЕТСЯ КАК ЕСТЬ, БЕЗ ЛЮБОГО ВИДА ГАРАНТИЙ, ЯВНО ВЫРАЖЕННЫХ ИЛИ ПОДРАЗУМЕВАЕМЫХ, ВКЛЮЧАЯ, НО НЕ ОГРАНИЧИВАЯСЬ ГАРАНТИЯМИ ТОВАРНОЙ ПРИГОДНОСТИ, СООТВЕТСТВИЯ ПО ЕГО КОНКРЕТНОМУ НАЗНАЧЕНИЮ И НЕНАРУШЕНИЯ ПРАВ. НИ В КАКОМ СЛУЧАЕ АВТОРЫ ИЛИ ПРАВООБЛАДАТЕЛИ НЕ НЕСУТ ОТВЕТСТВЕННОСТИ ПО ИСКАМ О ВОЗМЕЩЕНИИ УЩЕРБА, УБЫТКОВ ИЛИ ДРУГИХ ТРЕБОВАНИЙ ПО ДЕЙСТВУЮЩИМ КОНТРАКТАМ, ДЕЛИКТАМ ИЛИ ИНОМУ, ВОЗНИКШИМ ИЗ, ИМЕЮЩИМ ПРИЧИНОЙ ИЛИ СВЯЗАННЫМ С ПРОГРАММНЫМ ОБЕСПЕЧЕНИЕМ ИЛИ ИСПОЛЬЗОВАНИЕМ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ИЛИ ИНЫМИ ДЕЙСТВИЯМИ С ПРОГРАММНЫМ ОБЕСПЕЧЕНИЕМ.

Index

Packages

Programs

Libraries

Scripts

Others