Linux From Scratch

Версия 6.8

Автор Gerard Beekmans
Редакторы Matthew Burgess и Bruce Dubbs
Перевод Иван Лабутин

Linux From Scratch: Версия 6.8

by Автор Gerard Beekmans, Редакторы Matthew Burgess и Bruce Dubbs, Перевод Иван Лабутин

Содержание

Пролог

Предисловие

Мои приключения в изучении Linux начались больше десяти лет назад, в 1998. Я просто установил свой первый дистрибутив Linux и быстро стал поклонником концепции и философии Linux.

Всегда существуют несколько путей решения задачи. То же самое можно сказать о дистрибутивах Linux. Самые серьезные существуют годами. Некоторые все еще существуют, некоторые превратились во что-то другое, еще одни остались только в нашей памяти. Они все разные, все отражают потребности целевой аудитории. Поскольку существует такое огромное количество путей достижения одного результата, я начал понимать, что я более не обязан ограничиваться какой-либо одной реализацией. До исследования Linux, нам просто приходилось мириться с проблемами других операционных систем, поскольку у нас не было выбора. Это было так, нравилось Вам или нет. С Linux концепция выбора дошла до своего апогея. Если Вам что-то не нравится, Вы абсолютно свободно можете поменять это, настолько свободно, насколько вообще возможно.

Я попробовал несколько дистрибутивов и не смог остановиться ни на одном. Они все хороши, каждый по-своему. Более не существует понятий "правильно" и "неправильно". Теперь всем управляет Ваш личный вкус. При огромной свободе выбора я осознал, что невозможно подобрать себе одну идеальную во всем систему. Поэтому я решил создать мою собственную систему Linux, которая полностью бы соответствовала моим персональным предпчтениям.

Чтобы действительно получить свою собственную систему, я решил собирать абсолютно все из исходных кодов вместо того, чтобы использовать прекомпилированные бинарные пакеты. Эта «идеальная» Linux-система должна иметь сильные стороны всех других систем и исключать их слабости. Поначалу идея казалась весьма обескураживающей. Не верилось, что такая система может быть создана.

После долгого преодоления препятствий, таких как взаимные зависимости и ошибки компиляции, я наконец собрал свою собственную Linux-систему. Она была полностью готова для использования, как и любой другой дистрибутив Linux. Но это было мое творение. Это было необыкновенное чувство. Лучше этого было бы только самостоятельное написание каждого компонента системы.

Как только я поделился своими идеями с другими членами Linux-сообщества, стало ясно, что существует стойкий интерес к подобным проектам. Сразу стало ясно, что такие самосборные Linux-системы могут не только служить для удовлетворения специфических требований пользователя, но и быть идеальным обучающим материалом для программистов и системных администраторов, на котором они могли бы оттачивать свое мастерство. Именно из этих идей и родился проект *Linux From Scratch Project*.

Эта книга - ядро проекта Linux From Scratch. Она предоставляет фундамент и инструкции, необходимые Вам для построения и компиляции собственной системы. Эта книга дает шаблон, следуя которому Вы получите корректно работающую систему; Вы свободно можете изменять инструкции, чтобы результат соответствовал

Вашим желаниям, и на самом деле именно это и есть важнейшая часть проекта. Вы контролируете все; мы просто протягиваем руку помощи, чтобы помочь Вам в начале Вашего собственного приключения.

Я искренне надеюсь, что Вы замечательно проведете время, работая над своей собственной сборкой Linux From Scratch и наслаждаясь огромным числом преимуществ Своей Собственной Системы.

--
Gerard Beekmans
gerard@linuxfromscratch.org

Кому адресована эта книга?

Существует множество причин, по которым Вы могли захотеть прочесть эту книгу. Один из вопросов, который задают многие люди, «почему так необходимо полностью проходить весь процесс ручной сборки Linux-системы с нуля, когда можно просто скачать и установить уже готовый дистрибутив?»

Одна из важных целей существования этого проекта - помочь Вам изучить, как Linux-система работает изнутри. Сборка LFS помогает показать, что составляет Linux и как его компоненты взаимодействуют друг с другом. Одна из лучших вещей - это то, что приобретенный опыт самообучения поможет Вам в дальнейшем расширении Linux-системы в любом направлении.

Другой важный аспект LFS - это возможность полностью контролировать систему, не полагаясь при этом на чью-то-там реализацию дистрибутива Linux. С LFS, Вы находитесь в кресле водителя и диктуете каждый аспект своей системы.

LFS позволяет Вам создавать ультракомпактные Linux-системы. При установке обычного дистрибутива Вам часто приходится устанавливать огромное количество программ, которые никогда не будут использованы. Эти программы впустую занимают место на диске. Вы можете заметить, что с нынешнеми жесткими дисками это не так уж и страшно. Однако, иногда Вам будет важен размер системы. Вспомните о загрузочных CD, USB-дисках и встраиваемых системах. Это области, где LFS будет весьма выгоден.

Еще одна важная причина собственноручной сборки Linux - безопасность. Компилируя всю систему из исходного кода, Вы можете проверить все и применить необходимые патчи безопасности. Больше не нужно ждать, пока кто-нибудь другой откомпилирует бинарные пакеты и устранит в них дыру. Хотя Вы и можете проверить патч, нет никакой гарантии, что новый бинарный пакет был собран корректно и в нем действительно исправлена проблема.

Цель Linux From Scratch - собрать полную и готовую к использованию систему базового уровня. Если Вы не хотите собирать свою собственную Linux систему с нуля, Вам не удастся извлечь всей пользы из данной книги.

Конечно же, причин для сборки своей LFS-системы слишком много, чтобы перечислять здесь их все. Подводя итоги, знания являются самым весомым аргументом "за". Если Вы продолжите свое изучение LFS, Вы будуте поражены силой, которую дают информация и знания.

Целевые архитектуры LFS

Главная целевая архитектура LFS - 32-разрядные прцессоры Intel. Если Вы не собирали LFS до этого, Вам стоит начать именно с нее. 32-разрядная архитектура является самой широко распространенной и лучше остальных совместима и со свободным, и с проприентарным программным обеспечением.

Несмотря на это, инструкции в книге, с небольшими изменениями, работают и с Power PC, и с 64-разрядными процессорами AMD/Intel. Чтобы собрать систему, использующую один из этих процессоров, необходимо, в дополнение к другим нижеследующим требованиям, иметь существующую систему Linux, такую как более ранняя установка LFS, Ubuntu, Red Hat/Fedora, SuSE, которая также бы поддерживала данную архитектуру процессора. Также, помните, что 32-разрядный дистрибутив может быть установлен и использован как хост-система на 64-разрядных компьютерах AMD/Intel.

Необходимо сказать еще несколько слов о 64-разрядных системах. В сравнении с 32-разрядными, размер исполняемых файлов немного больше при практически незаметной разнице в скорости выполнения. Например, при тестовой сборке LFS-6.5 на системе с процессором Core2Duo были получены следующие результаты:

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

Как Вы можете видеть, 64-разрядная сборка только на 4% быстрее и при этом на 9% больше, чем 32-разрядная. Выгода от перехода на 64-разрядную систему крайне невелика. Конечно, если у Вас более 4GB RAM или Вам необходимо часто работать с данными, размер которых превышает 4GB, преимущества 64-разрядной системы очевидны.

По умолчанию, 64-разрядная система, которая получится при сборке LFS, является так называемой "чистой" 64-разрядной системой. Такая система поддерживает только 64-разрядные исполняемые файлы. Сборка "мульти-архитектурной" системы требует двойной компиляции многих приложений, один раз для 32-разрядных файлов и один раз для 64-разрядных. Это не поддерживается проектом LFS, поскольку не соотносится с идеей предоставления инструкций, необходимых для сборки простой Linux-системы. Вас может заинтересовать проект *Cross Linux From Scratch* ключевой целью которого и является сборка мультиархитектурной системы.

И напоследок еще одно замечание о 64-разрядных системах. Некоторые пакеты на данный момент не могут быть собраны для "чистой" 64-разрядной системы или требуют специальных инструкций по сборке. Как правило, такие приложения написаны с использованием специфичных для 32-разрядных систем ассемблерных инструкций, которые не позволяют собрать прграааму для 64-разрядной системы. Такими проблемными пакетами являются некоторые драйвера Xorg из *Beyond Linux From Scratch (BLFS)*. Большую часть таких проблем можно обойти, но это требует дополнительных специальных действий и патчей.

LFS и стандарты

Структура LFS следует стандартам Linux так строго, как только возможно. Главными стандартами являются:

- *The Single UNIX Specification Version 3 (POSIX)*. Примечание: Необходима регистрация (свободно).
- *Filesystem Hierarchy Standard (FHS)*
- *Linux Standard Base (LSB) Core Specification 4.0*

  LSB имеет пять отдельных частей: Core, C++, Desktop, Runtime Languages, и Printing. В дополнение к основным требованиям имеются архитектурно-специфичные. LFS старается следовать вышеприведенным правилам.

Замечание

Многие люди не согласны с требованиями LSB. Основной причиной их определения была необходимость в уверенности, что проприентарное программное обеспечение можно будет установить и нормально использовать на совместимой системе. Поскольку LFS - source-based система, пользователь имеет полный контроль над всеми пакетами и может отказаться от установки некоторых пакетов, требуемых по спецификациям LSB.

Создание LFS-системы, соответствующей всем спецификациям LSB, вполне возможно, но потребует установки множества дополнительных пакетов, которые находятся за пределами рассморения LFS. Большую часть из этих пакетов можно установить по инструкциям из BLFS.

Пакеты, предоставляемые LFS и необходимые для удовлетворения требований LSB

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

Пакеты, предоставляемые BLFS и необходимые для удовлетворения требований LSB

LSB Core:	Bc, Cpio, Ed, Fcrontab, PAM, Sendmail (или Postfix, или Exim)
LSB C++:	Нет
LSB Desktop:	ATK, Cairo, Desktop-file-utils, Freetype, Fontconfig, Glib2, GTK+2, Icon-naming-utils, Libjpeg, Libpng, Libxml2, MesaLib, Pango, Qt3, Qt4, Xorg
LSB Runtime Languages:	Python
LSB Printing:	CUPS
LSB Multimeda:	Alsa Libraries, NSPR, NSS, OpenSSL, Java

Пакеты, не предоставляемые LFS или BLFS и необходимые для удовлетворения требований LSB

LSB Core:	At, Batch, Install_initd, Lsb_release, Remove_initd, Test
LSB C++:	Нет
LSB Desktop:	Нет
LSB Runtime Languages:	Нет
LSB Printing:	Нет
LSB Multimeda:	Xdg-utils

Пояснения к выбранным пакетам

Как было сказано выше, цель проекта LFS - построение полной и готовой к использованию системы базового уровня. Она должна включать в себя все пакеты, необходимые для самовоспроизведения, предоставляя относительно небольшую основу, от которой пользователь может отталкиваться в построении своей, более сложной системы. Это не значит, что LFS не может быть сделана еще более маленькой. Некоторые важные пакеты, несмотря на наши рекомендации, могут быть более-менее безболезненно исключены из построения. Список, представленный ниже, поясняет роль каждого пакета в системе и причины, по которым он был включен в книгу.

- Autoconf

  Этот пакет содержит программы, создающие скрипты оболочки, способные автоматически сконфигурировать исходные коды из шаблона, предоставленного разработчиком. Он часто необходим для повторной сборки пакета после изменений в процедуре построения.

- Automake

  Пакет содержит программы для генерации Make-файлов из шаблонов. Он часто необходим для повторной сборки пакета после изменений в процедуре построения.

- Bash

  Этот пакет удовлетворяет требование LSB Core, по которому система должна предоставлять интерфейс Bourne Shell. Он был выбран из большого числа других вариантов потому, что является наиболее популярным и одним из самых мощных по возможностям.

- Binutils

  Этот пакет содержит компоновщик, ассемблер и другие утилиты для работы с объектными файлами. Программы из этого пакета необходимы для компиляции почти всех пакетов LFS и большинства остальных программ.

- Bison

  Пакет содержит GNU-версию yacc (Yet Another Compiler Compiler, Еще Один Компилятор Компиляторов), необходимого для сборки некоторых других программ LFS.

- Bzip2

Этот пакет содержит программы для работы со сжатыми данными. Он необходим для распаковки многих пакетов LFS.

- Coreutils

  Пакет включает в себя необходимые программы для просмотра и обработки файлов и каталогов. Они требуются для управления файлами из командной строки, а также для установки абсолютно всех пакетов LFS.

- DejaGNU

  Содержит компоненты для тестирования других программ. Этот пакет устанавливается только как временный инструментарий.

- Diffutils

  Пакет содержит программы, которые позволяют выявить различия между файлами или каталогами. С их помощью можно создавать патчи, а также они необходимы для сборки многих пакетов.

- Expect

  Этот пакет содержит программу для связывания скриптовых диалогов с другими интерактивными программами. Он зачастую используется при тестировании других пакетов. Он устанавливается только как временный инструментарий.

- E2fsprogs

  Пакет включает в себя утилиты для оперирования с файловыми системами ext2, ext3 и ext4. Это самые популярные и тщательно протестированные файловые системы, поддержимаемые ядром Linux.

- File

  Этот пакет включает в себя утилиту для определения типа переданного ей файла или нескольких файлов. Некоторые пакеты требуют ее для сборки.

- Findutils

  Пакет содержит программы для поиска файлов в файловой системе. Очень многие пакеты используют эти утилиты при сборке.

- Flex

  Этот пакет содержит утилиту для генерации программ, способных распознавать шаблоны в тексте. Это GNU-версия lex (лексического анализатора), необходимого для сборки некоторых пакетов LFS.

- Gawk

  В этом пакете содержится программа для оперирования содержимым текстовых файлов. Это GNU-версия awk (Aho-Weinberg-Kernighan), который используется в скриптах сборки многих пакетов.

- Gcc

  Это - Собрание Компиляторов GNU (Gnu Compiler Collection). Данный пакет содержит компиляторы C и C++, а также многие другие, не устанавливаемые в процессе сборки LFS.

- GDBM

  Пакет предоставляет библиотеку управления базами данных GNU (GNU Database Manager). Она используется другим пакетом, Man-DB.

- Gettext

  В данном пакете находятся утилиты для интернационализации и перевода интерфейса программ на другие языки. Они необходимы некоторым пакетам.

- Glibc

  Этот пакет содержит главную библиотеку языка C. Ни одна программа в Linux не запустится без нее.

- GMP

  This package contains math libraries and provide functions for arbitrary precision arithmetic. It is required to build Gcc.

- Grep

  This package contains programs for searching through files. These programs are used by most packages' build scripts.

- Groff

  This package contains programs for processing and formatting text. One important function of these programs is to format man pages.

- GRUB

  This package is the Grand Unified Boot Loader. It is one of several boot loaders available, but is the most flexible.

- Gzip

  This package contains programs for compressing and decompressing files. It is needed to decompress many packages in LFS and beyond.

- Iana-etc

  This package provides data for network services and protocols. It is needed to enable proper networking capabilities.

- Inetutils

  This package contains programs for basic network administration.

- IProute2

  This package contains programs for basic and advanced IPv4 and IPv6 networking. It was chosen over the other common network tools package (net-tools) for its IPv6 capabilities.

- Kbd

  This package contains key-table files, keyboard utilities for non-US keyboards, and a number of console fonts.

- Less

  This package contains a very nice text file viewer that allows scrolling up or down when viewing a file. It is also used by Man-DB for viewing manpages.

- Libtool

  This package contains the GNU generic library support script. It wraps the complexity of using shared libraries in a consistent, portable interface. It is needed by the test suites in other LFS packages.

- Linux Kernel

  This package is the Operating System. It is the Linux in the GNU/Linux environment.

- M4

  This package contains a general text macro processor useful as a build tool for other programs.

- Make

  This package contains a program for directing the building of packages. It is required by almost every package in LFS.

- Man-DB

  This package contains programs for finding and viewing man pages. It was chosen instead of the man package due to superior internationalization capabilities. It supplies the man program.

- Man-pages

  This package contains the actual contents of the basic Linux man pages.

- MPC

  This package contains functions for the arithmetic of complex numbers. It is required by Gcc.

- Module-Init-Tools

  This package contains programs needed to administer Linux kernel modules.

- MPFR

  This package contains functions for multiple precision arithmetic. It is required by Gcc.

- Ncurses

  This package contains libraries for terminal-independent handling of character screens. It is often used to provide cursor control for a menuing system. It is needed by a number of packages in LFS.

- Patch

  This package contains a program for modifying or creating files by applying a *patch* file typically created by the diff program. It is needed by the build procedure for several LFS packages.

- Perl

  This package is an interpreter for the runtime language PERL. It is needed for the installation and test suites of several LFS packages.

- Pkg-config

  This package contains a tool for passing the include path and/or library paths to build tools during the configure and make processes. It is needed by many LFS packages.

- Procps

  Пакет содержит программы для слежения за работой процессов. Эти программы полезны для администрирования системы, а также используются загрузочными скриптами LFS.

- Psmisc

  Пакет предоставляет программы, выводящие различную информацию о запущенных процессах. Они полезны для системного администрирования.

- Readline

  Пакет содержит набор библиотек, предоставляющих возможность редактирования командной строки и хранения истории команд. Он используется Bash.

- Sed

  Этот пакет позволяет редактировать текст без открытия его в текстовом редакторе. Он также требуется большинством конфигурационных скриптов.

- Shadow

  Пакет содержит программы для безопасного управления паролями.

- Sysklogd

  Этот пакет содержит программы для журналирования системных сообщений, подобных тем, что ядро или демоны посылают в случае необычного события.

- Sysvinit

  В этом пакете содержится программа init, являющаяся родителем всех остальных процессов в системе Linux.

- Tar

  Этот пакет предоставляет возможность создания архивов и их распаковки. Необходим для извлечения абсолютно всех пакетов, используемых в LFS.

- Tcl

  Пакет содержит Tool Command Language, используемый при выполнении тестирования во многих пакетах LFS. Он устанавливается только как временный инструментарий.

- Texinfo

  Этот пакет содержит программы для чтения, создания и преобразования info-страниц. Он используется при установке многих пакетов LFS.

- Udev

  Пакет содержит программы для динамической генерации узлов устройств. Udev является альтернативой созданию нескольких тысяч статических устройств в директории /dev.

- Util-linux

  Пакет включает в себя разнообразные утилиты. Среди них программы для управления файловыми системами, разделами, консолью и сообщениями.

- Vim

  Этот пакет содержит редактор. Он был выбран из-за совместимости с класическим редактором vi и огромного числа мощных возможностей. Выбор редактора - очень субъективный момент, поэтому Вы, по желанию, пожете заменить Vim любым другим текстовым редактором.

- XZ Utils

Данный пакет включает в себя программы для сжатия и распаковки файлов. На данный момент, они предоставляют наилучшее вообще возможное сжатие, и необходимы для распаковки пакетов формата XZ или LZMA.

- Zlib

  Пакет содержит библиотеку процедур компрессии/декомпрессии, используемую некоторыми программами.

Необходимые знания

Сборка системы LFS - непростая задача. Она требует некоторого умения администрировать Unix-системы, чтобы решать возникающие проблемы и правильно выполнять написанные команды. Как абсолютный минимум, Вы уже должны уметь использовать командную строку (оболочку): копировать или перемещать файлы и папки, просматривать содержимое папок и файлов, менять текущую рабочую директорию. Также Вы должны знать, как устанавливать и использовать программное обеспечение в Linux.

Поскольку книга LFS предполагает *как минимум* наличия этих базовых умений, различные форумы поддержки LFS не предоставят Вам помощь по таким вопросам. Вы будете расстроены, что Ваши просьбы помочь с основными навыками либо останутся вообще без ответа, либо ответы будут содержать лишь ссылки на эту страницу - не лучше ли будет прочитать эту литературу сразу?

Перед сборкой LFS системы мы рекомендуем прочитать следующие HOWTO:

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

  Это замечательное руководство по сборке и установке «основных» пакетов программного обеспечения Unix под Linux. Хотя оно и было написано достаточно давно, оно до сих пор позволит получить основные навыки, необходимые для сборки и установки программного обеспечения.

- The Linux Users' Guide *http://www.linuxhq.com/guides/LUG/guide.html*

  Это руководство рассказывает об использовании различного программного обеспечения Linux. Оно также очень старое, но своей актуальности не утратило.

- The Essential Pre-Reading Hint *http://www.linuxfromscratch.org/hints/downloads/files/essential_prereading.txt*

  Это LFS Hint, написанный специально для новичков в Linux. Он включает в себя список ссылок на великолепные источники информации по большому кругу различных тем. Любой, кто хочет установить LFS, должен понимать большую часть из них.

Требования к хост-системе

Данный список содержит имена и минимальные версии пакетов, которые необходимо установить на Вашей хост-системе. Это не должно быть проблемой для последних дистрибутивов Linux. Не забывайте, что многие дистрибутивы помещают заголовочные файлы в отдельные пакеты, часто в виде «<package-name>-devel» или «<package-name>-dev». Вам необходимо установить и их, если Ваш дистрибутив их предоставляет.

Более старые версии перечисленных пакетов могут работать, но корректность их работы не проверялась.

- **Bash-3.2** (/bin/sh должен быть символической ссылкой на bash)
- **Binutils-2.17** (Версии новее, чем 2.21 не рекомендуются, поскольку они не были протестированы)
- **Bison-2.3** (/usr/bin/yacc должен быть ссылкой на bison или маленьким скриптом, запускающим bison)
- **Bzip2-1.0.4**
- **Coreutils-6.9**
- **Diffutils-2.8.1**
- **Findutils-4.2.31**
- **Gawk-3.1.5** (/usr/bin/awk должен быть ссылкой на gawk)
- **Gcc-4.1.2** (Версии новее, чем 4.5.2 не рекомендуются, поскольку они не были протестированы)
- **Glibc-2.5.1** (Версии новее, чем 2.13 не рекомендуются, поскольку они не были протестированы)
- **Grep-2.5.1a**
- **Gzip-1.3.12**
- **Linux Kernel-2.6.22.5** (откомпилированное GCC-4.1.2 или более новым)

  Причина, по которой необходимо ограничение на версию ядра кроется в том, что мы указываем эту версию при сборке glibc в Главе 6, как рекомендуется разработчиками.

  Если ядро хост-системы не 2.6.22.5, или если оно было собрано не компилятором GCC-4.1.2 (или более поздним), Вам необходимо заменить ядро на соответствующее этим требованиям. Существуют два способа сделать это. Во-первых, проверьте, не предоставляет ли Ваш Linux дистрибутив ядро 2.6.22.5 или более новое. Если это так, Вам следует установить его. Если же Ваш дистрибутив не предоставляет приемлемое ядро, или Вы не хотите устанавливать его, Вы можете собрать ядро самостоятельно. Инструкции по сборке ядра и конфигурированию загрузчика (предполагая, что хост-система использует GRUB), расположены здесь: Chapter 8.
- **M4-1.4.10**
- **Make-3.81**
- **Patch-2.5.4**
- **Perl-5.8.8**
- **Sed-4.1.5**
- **Tar-1.18**
- **Texinfo-4.9**

Заметьте, что символические ссылки, описанные выше, необходимы для сборки LFS по инструкциям этой книги. Символические ссылки, указывающие на другие приложения (вроде dash, mawk, и т.д.) могут работать, но не были проверены и не поддерживаются командой разработки LFS, и могут потребовать отклонений от инструкций или дополнительных патчей к некоторым пакетам.

Чтобы проверить, установлено ли на Вашей системе все необходимое и можно ли в ней компилировать программы, запустите следующий скрипт:

```
cat > version-check.sh << "EOF"
#!/bin/bash
export LC_ALL=C

# Simple script to list version numbers of critical development tools

bash --version | head -n1 | cut -d" " -f2-4
echo "/bin/sh -> `readlink -f /bin/sh`"
echo -n "Binutils: "; ld --version | head -n1 | cut -d" " -f3-
bison --version | head -n1
if [ -e /usr/bin/yacc ];
  then echo "/usr/bin/yacc -> `readlink -f /usr/bin/yacc`";
  else echo "yacc not found"; fi
bzip2 --version 2>&1 < /dev/null | head -n1 | cut -d" " -f1,6-
echo -n "Coreutils: "; chown --version | head -n1 | cut -d")" -f2
diff --version | head -n1
find --version | head -n1
gawk --version | head -n1
if [ -e /usr/bin/awk ];
  then echo "/usr/bin/awk -> `readlink -f /usr/bin/awk`";
  else echo "awk not found"; fi
gcc --version | head -n1
/lib/libc.so.6 | head -n1 | cut -d"," -f1
grep --version | head -n1
gzip --version | head -n1
cat /proc/version
m4 --version | head -n1
make --version | head -n1
patch --version | head -n1
echo Perl `perl -V:version`
sed --version | head -n1
tar --version | head -n1
echo "Texinfo: `makeinfo --version | head -n1`"
echo 'main(){}' > dummy.c && gcc -o dummy dummy.c
if [ -x dummy ]; then echo "Compilation OK";
  else echo "Compilation failed"; fi
rm -f dummy.c dummy

EOF

bash version-check.sh
```

Соглашения, используемые в книге

Чтобы Вам легче было следовать инструкциям, здесь приводится разъяснение некоторых обозначений, используемых в книге Linux From Scratch.

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

Текст такого типа необходимо вводить так, как он напечатан, если явно не сказано иное. Также он будет использован в секциях объяснения, для подчеркивания вызываемой команды.

В некоторых случаях одна логическая строка разделена на две или более физические строки с помощью обратного слеша в конце строки.

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

Заметьте, что за обратным слешем сразу же идет перевод строки. Другие символы пробелов или табуляций приведут к неверным результатам.

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

Текст такого типа (моноширинный текст) показывает вывод с экрана, обычно как результат выполненной команды. Этот формат также используется для выделения имен файлов, например `/etc/ld.so.conf`.

Курсив

Этот тип текста используется в книге для нескольких целей, в основном для выделения важных моментов и понятий.

http://www.linuxfromscratch.org/

Такой формат используется для гиперссылок, включающих ресурсы сообщества LFS, HOWTO, зеркала и другие веб-сайты.

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

Этот формат используется при создании конфигурационных файлов. Первая команда говорит системе создать файл `$LFS/etc/group` из того, что вводится в следующих строках вплоть до последовательности конца файла (End Of File, EOF). Эта секция чаще всего вводится как она есть.

<ЗАМЕНЯЕМЫЙ ТЕКСТ>

Такой формат используется для участков текста, которые не должны вводиться как есть или копироваться и вставляться.

[НЕОБЯЗАТЕЛЬНЫЙ ТЕКСТ]

Этот формат используется для участков текста, которые не являются обязательными.

`passwd(5)`

Такой тип текста используется для указания на определенную страницу руководства (man). Номер внутри скобок указывает секцию, в которой находится страница. Например, у **passwd** есть две страницы руководства. В соответствии с инструкциями LFS, эти две страницы будут расположены в `/usr/share/man/man1/passwd.1` и `/usr/share/man/man5/passwd.5`. Когда книга использует `passwd(5)`, это означает специальную отсылку к `/usr/share/man/man5/passwd.5`. **man passwd** напечатает

первую найденную страницу руководства с именем «passwd», которой будет `/usr/share/man/man1/passwd.1`. В этом примере Вам необходимо выполнить **man 5 passwd** чтобы прочесть нужную страницу. Нужно отметить, что большинство страниц руководства не имеют дублирующихся имен в разных секциях. Поэтому **man *<program name>***, скорее всего, подойдет в большинстве случаев.

Структура

Эта книга разделена на следующие части.

Часть I - Введение

Часть I объясняет несколько важных моментов, необходимых для установки LFS. Эта секция также содержит информацию о самой книге.

Часть II - Подготовка к сборке

Часть II описывает подготовку к процессу сборки—создание раздела, загрузка пакетов и компиляция временного инструментария.

Часть III - Сборка системы LFS

Часть III проводит читателя через процесс сборки LFS системы— компиляция и установка всех пакетов по одному, настройка загрузочных скриптов и установка ядра. Полученная Linux-система является хорошей базой, на которую можно установить дополнительное программное обеспечение, чтобы расширить систему как угодно. В конце книги приведен легкий в использовании глоссарий всех установленных программ, библиотек и важных файлов.

Предупреждения об ошибках

Программное обеспечение, используемое для создания LFS, постоянно обновляется и улучшается. Предупреждения безопасности и исправления ошибок могут появиться после выхода книги о LFS. Чтобы проверить, нужно ли обновить версию какого-либо пакета или инструкции его по сборке в связи с найденными уязвимостями и другими ошибками, пожалуйста, посетите *http://www.linuxfromscratch.org/lfs/errata/6.8/* перед началом сборки системы. Необходимо внимательно изучить все изменения и аккуратно применить их к соответствующей части книги на соответствующем этапе ее построения.

Часть I. Введение

Глава 1. Введение

1.1. Как собрать LFS-систему

Система LFS будет построена с использованием уже установленного дистрибутива Linux (такого как Debian, Mandriva, Red Hat или Suse). Эта существующая Linux-система (хост) будет использована как отправная точка, предоставляющая необходимые программы - компилятор, компоновщик и оболочку - для сборки новой системы. Выберите компонент «Разработка» при установке дистрибутива, чтобы иметь доступ к этим инструментам.

Вместо того, чтобы устанавливать отдельный дистрибутив на жесткий диск Вашего компьютера, Вы можете использовать Linux From Scratch LiveCD или LiveCD другого дистрибутива. К сожалению, в последнее время разработка LiveCD приостановлена, и он содержит устаревшие версии как пакетов и патчей (версии, помеченные «-nosrc» или «-min» не содержат их вообще), так и этой книги. Если Вы заинтересовались LFS LiveCD и/или хотите скачать копию, пожалуйста, посетите *http://www.linuxfromscratch.org/livecd/*.

Замечание

LFS LiveCD может не работать на новых компьютерах, не запускаясь или не определяя некоторое оборудование, в частности, некоторые жесткие диски SATA.

Глава 2 этой книги описывает, как создать новый родной Linux-раздел и файловую систему на нем. Это место, где система LFS будет собрана и установлена. Глава 3 объясняет, какие пакеты и патчи необходимо загрузить, чтобы построить LFS-систему, и как сохранить их на новой файловой системе. Глава 4 рассказывает об установке правильного рабочего окружения. Пожалуйста, внимательно прочтите Глава 4, поскольку в ней говорится о нескольких важных моментах, кокторые Вам необходимо знать, прежде чем переходить к Глава 5 и далее.

Глава 5 описывает установку пакетов, формирующих базовую среду разработки (или набор инструментов), с помощью которых будет построена новая система в Chapter 6. Некоторые из этих пакетов необходимы для разрешения циклических зависимостей —например, чтобы скомпилировать компилятор, Вам необходим компилятор.

Глава 5 также расскажет Вам о том, как начерновую установить инструментарий, включая Binutils и GCC (начерновую означает, что далее эти два ключевых пакета будут переустановлены). Следующим шагом будет сборка Glibc, библиотеки языка C. Glibc будет скомпилирована с помощью программ, собранных на предыдущем шаге. Затем инструментарий будет пересобран для того, чтобы связать его динамически с только что скомпилированной Glibc. Следующие пакеты в Глава 5 будут собраны с использованием этого инструментария. Когда это будет сделано, процесс сборки LFS больше не будет зависеть от хост-системы, исключая, конечно, запущенное ядро.

Усилия по тщательной изоляции новой системы от хост-дистрибутива могут показаться излишними. Полное техническое разъяснение причин, по которым это делается, представлены в Раздел 5.2, «Замечания о методе сборки».

В Chapter 6 система LFS будет полностью собрана. Команда **chroot** (change root, смена корня) будет использована для входа в виртуальное окружение и запуска новой оболочки, чьей корневой директорией будет установлен раздел LFS. Это очень

похоже на перезагрузку и указание ядру использовать раздел LFS как корневой. Система не перезагружается, вместо этого используется **chroot**, так как создание загружаемой системы требует дополнительных усилий, которые на данном этапе не нужны. Главный плюс использования **chroot** - это то, что Вы можете спокойно продолжать пользоваться Вашей хост-системой, пока LFS собирается. Пока Вы ждете окончания компиляции пакета, Вы можете продолжать использовать Ваш компьютер как обычно.

Для завершения установки в Глава 7 будут установлены скрипты загрузки LFS-Bootscripts, а в Chapter 8 - ядро и загрузчик. Глава 9 содержит информацию по продолжению изучения LFS за пределами этой книги. После выполнения всех шагов, описанных здесь, компьютер наконец будет готов для перезагрузки в новую LFS-систему.

Это весь процесс в двух словах. Подробная информация о каждом шаге будет постепенно раскрываться в следующих главах и в описаниях пакетов. Вещи, кажущиеся сложными, буут подробно объяснены, и все будет раскладываться по полочкам по мере вашего погружения в увлекательное приключение LFS.

1.2. Обновления

Ниже приведен список пакетов и патчей, обновленных со времени прошлшго выпуска книги.

Обновление до:

- Autoconf 2.68
- Bash 4.2
- Binutils 2.21
- Bzip2 1.0.6
- Coreutils 8.10
- E2fsprogs 1.41.14
- Expect 5.45
- File 5.05
- GCC 4.5.2
- Glibc 2.13
- Grep 2.7
- Groff 1.21
- IPRoute2 2.6.37
- Libtool 2.4
- Linux 2.6.37
- M4 1.4.15
- Man-DB 2.5.9
- Man-pages 3.32
- Perl 5.12.3
- Psmisc 22.13

- Readline 6.2
- Shadow 4.1.4.3
- Tar 1.25
- TCL 8.5.9
- Udev 166

Добавлены:

- bzip2-1.0.6-install_docs-1.patch
- coreutils-8.10-i18n-1.patch
- coreutils-8.10-uname-1.patch
- gcc-4.5.2-startfiles_fix-1.patch
- glibc-2.13-gcc_fix-1.patch
- perl-5.12.3-libc-1.patch
- procps-3.2.8-fix_HZ_errors-1.patch
- xz-5.0.1
- util-linux-2.19

Удалены:

- bash-4.1-fixes-2.patch
- bzip2-1.0.5-install_docs-1.patch
- bzip2-1.0.5-version_fixes-1.patch
- coreutils-8.5-i18n-1.patch
- coreutils-8.5-uname-2.patch
- expect-5.44.1.15-no_tk-1.patch
- gcc-4.5.1-startfiles_fix-1.patch
- glibc-2.12.1-gcc_fix-1.patch
- glibc-2.12.1-makefile_fix-1.patch
- man-db-2.5.7-fix_man_assertion-1.patch
- perl-5.12.1-libc-1.patch
- tar-1.23-overflow_fix-1.patch
- util-linux-ng-2.18

1.3. Список изменений

Эта книга Linux From Scratch имеет версию 6.8 и была выпущена March 4, 2011. Если с момента выпуска прошло более шести месяцев, возможно, уже доступна более новая версия. Вы можете проверить ее наличие на одном из зеркал: *http://www.linuxfromscratch.org/mirrors.html*.

Ниже перечислены изменения, произошедшие в книге по сравнению с предыдущим выпуском.

Changelog Entries:

- 2011-03-04

    - [bdubbs] Release LFS 6.8.
- 2011-02-18
    - [bdubbs] Fix several urls in Chapter 3. Thanks to splotz90 for the patch.
    - [bryan] Fix the sed in the CD-ROM symlinks section, to sync with upstream changes to the file being modified.
- 2011-02-16
    - [matthew] Upgrade to Shadow-4.1.4.3. Fixes *#2832*.
    - [matthew] Upgrade to Readline-6.2. Fixes *#2831*.
    - [matthew] Upgrade to Bash-4.2. Fixes *#2830*.
    - [matthew] Upgrade to Udev-166. Fixes *#2829*.
- 2011-02-10
    - [bdubbs] Upgrade to coreutils-8.10. Fixes *#2828*.
    - [bdubbs] Upgrade to Util-linux-2.19. Changed name from util-linux-ng. Fixes *#2805*.
- 2011-02-04
    - [matthew] Upgrade to Glibc-2.13. Fixes *#2827*.
    - [matthew] Upgrade to XZ-5.0.1. Fixes *#2826*.
    - [matthew] Upgrade to Perl-5.12.3. Fixes *#2824*.
- 2011-01-27
    - [bdubbs] Add a sed that modifies incorrect defines in glibc. Thanks to Bryan Kadzban for identifying the proper fix. Fixes *#2820*.
- 2011-01-25
    - [bdubbs] Add a note about optionally building popt before pkg-config. Fixes *#2781*.
- 2011-01-24
    - [bdubbs] Move chroot man page to man8. Fixes *#2782*.
- 2011-01-23
    - [matthew] Ignore failing tests in Man-DB as they're only due to a change in Groff-1.21's warning output. Fixes *#2823*.
    - [matthew] Change password hashing from MD5 to SHA-512. Fixes *#2814*.
    - [matthew] Upgrade to File-5.05. Fixes *#2821*.
    - [matthew] Upgrade to IPRoute2-2.6.37. Fixes *#2817*.
    - [matthew] Upgrade to Coreutils-8.9. Fixes *#2815*.
- 2011-01-10
    - [ken] Updated to Linux-2.6.37. Fixes *#2816*.
    - [ken] Updated to Groff-1.21. Fixes *#2813*.
- 2011-01-04
    - [bdubbs] Move XZ-Utils to before Man-DB. Tweak install instructions.
- 2011-01-02
    - [bdubbs] Added XZ-Utils as a new compression utility package for independent use or with tar. Fixes *#2619*.

- 2010-12-29
  - [ken] Allow shadow to install its korean and chinese man-pages, since man-db can now format them. Thanks to William Immendorf for the report.
- 2010-12-28
  - [matthew] Upgrade to E2fsprogs-1.41.14. Fixes *#2812*.
  - [matthew] Upgrade to Coreutils-8.8. Fixes *#2811*.
- 2010-12-19
  - [matthew] Upgrade to Udev-165. Fixes *#2810*.
  - [matthew] Upgrade to GCC-4.5.2. Fixes *#2809*.
  - [matthew] Upgrade to E2fsprogs-1.41.13. Fixes *#2807*.
- 2010-12-14
  - [matthew] Upgrade to Glibc-2.12.2. Fixes *#2804*.
- 2010-12-13
  - [matthew] Upgrade to Binutils-2.21. Fixes *#2803*.
  - [matthew] Upgrade to Man-Pages-3.32. Fixes *#2802*.
  - [matthew] Upgrade to Linux-2.6.36.2. Fixes *#2799*.
  - [matthew] Upgrade to Man-DB-2.5.9. Fixes *#2797*.
- 2010-11-18
  - [matthew] Upgrade to Man-Pages-3.31. Fixes *#2794*.
  - [matthew] Upgrade to Expect-5.45. Fixes *#2791*.
- 2010-11-10
  - [matthew] Add security fixes for Glibc. Fixes *#2790*.
  - [matthew] Upgrade to Man-Pages-3.30. Fixes *#2788*.
  - [matthew] Mention Inetutils' testsuite, and also install its HTML documentation. Fixes *#2784* and *#2785*
  - [matthew] Upgrade to Tar-1.25, and also install its HTML documentation. Fixes *#2777* and *#2786*.
- 2010-10-27
  - [bdubbs] Add an example on how to use wget-list. Fixes *#2778*.
- 2010-10-26
  - [bdubbs] Clarify text in Chapter 5 GCC Pass 1 concerning supporting packages.
- 2010-10-24
  - [matthew] Upgrade to Udev-164. Fixes *#2775*.
  - [matthew] Upgrade to Man-Pages-3.29. Fixes *#2774*.
  - [matthew] Upgrade to Linux-2.6.36. Fixes *#2773*.
  - [matthew] Upgrade to Coreutils-8.6. Fixes *#2771*.
- 2010-10-18
  - [matthew] Upgrade to Bash 4.1 patch level 9. Fixes *#2770*.
  - [matthew] Upgrade to Udev-163. Fixes *#2769*.

- [matthew] Upgrade to Man-Pages-3.28. Fixes *#2765*.
- [matthew] Upgrade to Linux-2.6.35.7. Fixes *#2764*.
- [matthew] Upgrade to Autoconf-2.68. Fixes *#2763*.
- [matthew] Upgrade to Libtool-2.4. Fixes *#2762*.

- 2010-09-22
  - [matthew] Following r9370, fix the Autoconf underquoting bug in Autoconf itself, rather than just the one affected LFS package, Pkg-config.
  - [matthew] Upgrade to Linux-2.6.35.5. Fixes *#2761*.
  - [matthew] Upgrade to Grep-2.7. Fixes *#2760*.
  - [matthew] Upgrade to Bzip2-1.0.6. Fixes *#2759*.
  - [matthew] Add patch to fix "Unknown HZ value" error in some procps tools. Thanks to DJ Lucas for the report and patch. Fixes *#2758*.
  - [matthew] Upgrade to Tcl-8.5.9. Fixes *#2753*.
  - [matthew] Upgrade to Perl-5.12.2. Fixes *#2752*.
  - [matthew] Upgrade to Psmisc-22.13. Fixes *#2751*.
  - [matthew] Upgrade to Man-Pages-3.27. Fixes *#2750*.
  - [matthew] Upgrade to Udev-162. Fixes *#2747*.
  - [matthew] Upgrade to M4-1.4.15. Fixes *#2744*.
- 2010-09-18
  - [bdubbs] - LFS-6.7 released.

1.4. Ресурсы

1.4.1. FAQ

Если во время сборки LFS Вы получаете ошибки, у Вас есть вопросы, или считаете, что в книге опечатка, пожалуйста, ознакомьтесь со списком Frequently Asked Questions (FAQ), который находятся по адресу *http://www.linuxfromscratch.org/faq/*.

1.4.2. Списки рассылки

Сервер `linuxfromscratch.org` поддерживает несколько списков рассылки, используемых для разработки проекта LFS. Это списки главной разработки и поддержки, а также некоторые другие. Если FAQ не решил Вашей проблемы, следующим шагом должен стать поиск по спискам рассылки на *http://www.linuxfromscratch.org/search.html*.

Информация по различным спискам, правила подписки, расположение архивов и многое другое доступно здесь: *http://www.linuxfromscratch.org/mail.html*.

1.4.3. IRC

Несколько человек из команды разработки LFS поддерживают сервер Internet Relay Chat (IRC). Перед тем, как обратиться туда, убедитесь, пожалуйста, что Ваш вопрос уже не отвечен в LFS FAQ или в архивах списков рассылки. Сервер расположен по адресу `irc.linuxfromscratch.org`. Канал поддержки называется #LFS-support.

1.4.4. Зеркала

Проект LFS имеет множество зеркал по всему миру, чтобы сделать сделать доступ к веб-сайту и загрузку необходимых пакетов более удобными. Пожалуйста, посетите веб-сайт LFS *http://www.linuxfromscratch.org/mirrors.html*, чтобы получить список текущих активных зеркал.

1.4.5. Контактная информация

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

1.5. Помощь

Если у Вас возникли проблемы или какие-то вопросы при работе с этой книгой, пожалуйста, проверьте страницу FAQ *http://www.linuxfromscratch.org/faq/#generalfaq*. Вопрос, скорее всего, уже отвечен там. Если Вашего вопроса нет на этой странице, постарайтесь найти источник проблемы. Следующий совет поможет Вам в поиске правильного направления при решении проблемы: *http://www.linuxfromscratch.org/hints/downloads/files/errors.txt*.

Если Вы не нашли своей проблемы в FAQ, поищите решение в списках рассылки *http://www.linuxfromscratch.org/search.html*.

LFS также имеет многочисленное сообщество, готовое предложить Вам свою помощь через списки рассылки и IRC (см. Раздел 1.4, «Ресурсы»). К сожалению, мы получаем некоторые вопросы каждый день, и большинство из них может быть легко решено прочтением FAQ и поиском по спискам рассылки. Пожалуйста, прежде чем обращаться к нам, попробуйте сами исследовать проблему. Это поможет нам сконцентрироваться на действительно серьезных и сложных вопросах. Если Вы все же так и не нашли решения, пожалуйста включите в Ваш запрос к нам всю необходимую информацию.

1.5.1. Необходимые данные

Кроме краткого описания проблемы, вот необходимые моменты, которые должны быть в Вашем запросе:

- Версия используемой книги (в данном случае 6.8)
- Хост-дистрибутив и его версия, используемая для сборки LFS
- Вывод скрипта Раздел vii, «Требования к хост-системе» [xix]
- Название пакета или секции, где возникла проблема
- Точное сообщение об ошибке или ее точное описание
- Замечание, где Вы отклонялись от книги и как

Замечание

То, что Вы отклонялись от действий, описанных в книге, *не* значит, что мы не станем Вам помогать. В конце концов, LFS - это Ваша система. Ваше сообщение о изменениях в процедуре сборки поможет нам оценить и вычислить возможные причины Вашей пролблемы.

1.5.2. Проблемы при выполнении configure

Если что-то происходит не так при выполнении скрипта **configure**, просмотрите файл `config.log`. Он может содержать ошибки, произошедшие во время **configure**, которые не были выведены на экран. Включите *уместные* строки, если просите помощи.

1.5.3. Проблемы компиляции

Как вывод команды на экран, так и содержимое различных файлов может быть полезным при определении источника ошибок компиляции. Вывод скрипта **configure** и запуска **make** может быть уместным. Совсем не стоит включать полный вывод, выберите только необходимую и уместную информацию. Ниже пример части из вывода **make**, которую нужно включить в просьбу о помощи:

```
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 подобном случае большинство включат в свое сообщение только последнюю строку:

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

Этой информации недостаточно, чтобы корректно идентифицировать проблему, поскольку она просто говорит, что что-то пошло не так, а не *что* пошло не так. В данном примере необходимо включать в свое сообщение всю секцию, так как она включает в себя выполненную команду и связанные с ней ошибки.

Великолепная статья о том, как правильно просить помощи в Интернет, доступна здесь: *http://catb.org/~esr/faqs/smart-questions.html*. Прочитайте ее и следуйте советам, это увеличит Ваши шансы на получение помощи.

Часть II. Подготовка к сборке

Глава 2. Подготовка нового раздела

2.1. Вступление

В этой главе будет подготовлен раздел под будущую LFS-систему. Мы создадим сам раздел, файловую систему на нем и примонтируем его.

2.2. Создание нового раздела

Как и большинство других операционных систем, LFS обычно устанавливается на отдельный раздел жесткого диска. Рекомендуемый подход при сборке LFS-системы - использовать доступный свободный раздел, или, если у Вас достаточно неразмеченного пространства, создать новый.

Минимальная система требует раздела размером около 1.3 гигабайт (GB). Этого должно хватить для сохранения всех архивов с исходными текстами и компиляции пакетов. Однако, если Вы собираетесь использовать LFS как основную систему, скорее всего Вы будете устанавливать дополнительное программное обеспечение, которое потребует дополнительного места на диске (2-3 GB). LFS-система не будет полностью занимать все это место. Большая часть из требуемого необходима для предоставления свободного временного хранилища. Компиляция пакета может потребовать огромного свободного места на диске, которое будет освобождено после его установки.

Поскольку оперативной памяти не всегда может быть достаточно для процесса компиляции, хорошей идеей будет использовать небольшой раздел диска как раздел подкачки. Он используется ядром для сохранения редко используемых данных, выгрузка которых из оперативной памяти позволяет выделить больше места в ней для активных процессов. LFS-система может использовать тот же раздел подкачки, что и хост-система, в этом случае не обязательно создавать новый.

Запустите программу разметки диска, например **cfdisk** или **fdisk**, и передайте ей в параметрах имя жесткого диска, на котором хотите создать раздел—например, /dev/hda для первичного Integrated Drive Electronics (IDE) диска. Создайте родной Linux-раздел и раздел подкачки, если необходимо. Пожалуйста, прочтите cfdisk(8) или fdisk(8) если не знаете, как пользоваться этими программами.

Запомните обозначение нового раздела (например, hda5). В этой книге он будет подразумеваться под разделом LFS. Также запомните обозначение раздела подкачки. Эти имена будут необходимы в дальнейшем, в том числе и для файла /etc/fstab.

2.2.1. Вопросы разметки диска

Просьбы помочь с разметкой диска часто встречаются в списках рассылки LFS. Это весьма субъективный вопрос. По умолчанию большинство дистрибутивов используют весь диск за исключением одного небольшого раздела подкачки. Это не является оптимальным для LFS по нескольким причинам. Это уменьшает гибкость, делает совместное использование данных между несколькими дистрибутивами или сборками LFS более сложным и затрудняет возможность резервного копирования.

2.2.1.1. Корневой раздел

Корневой раздел LFS (не перепутайте с директорией `/root`) размером около 10 гигабайт должен быть хорошим компромиссом для большинства систем. Этого будет достаточно для сборки LFS и большей части BLFS, но останется еще место, чтобы создать несколько разделов для экспериментов.

2.2.1.2. Раздел подкачки

Большинство дистрибутивов автоматически создают раздел подкачки. В большинстве случаев рекомендуемый размер раздела - удвоенный объем оперативной памяти, хотя вряд ли Вам понадобится столько. Если место на диске ограничено, сделайте раздел подкачки размером в два гигабайта и следите за процессом подкачивания

Подкачка - это плохо. Обычно Вы можете понять, что система включила механизм подкачки, просто слыша активную работу диска и замечая, как ситема реагирует на Ваши действия. Первым делом в такой ситуации необходимо проверить, не была ли введена неверная команда, например запрос на редактирование гигабайтного файла. Если подкачка становится нормальным поведением, лучшим решением будет прикупить больше оперативной памяти для системы.

2.2.1.3. Полезные разделы

Можно создать еще несколько других разделов, которые не являются обязательными, но о них стоит задуматься, планируя разметку диска. Следующий список не является всеобъемлющим, но вполне может рассматриваться как руководство.

- /boot – Весьма рекомендуется. На этом разделе можно хранить ядра и другую загрузочную информацию. Чтобы минимизировать потенциальные проблемы, связанные с загрузкой с больших дисков, сделайте этот раздел первичным и расположьте его в начале Вашего первого жесткого диска. Вполне достаточно будет выделить под него около 100 мегабайт.
- /home – Весьма рекомендуется. Стоит использовать один домашний раздел для нескольких дистрибутивов или установленных сборок LFS. Размер обычно очень большой, выделите под него все возможное доступное место.
- /usr – Отдельный раздел /usr обычно используется в конфигурации с сервером, управляющим тонкими клиентами или бездисковыми рабочими станциями. Он не является необходимым для LFS. Размера в пять гигабайт должно хватить для большинства установок.
- /opt – Эта директория будет очень полезна для BLFS. Некоторые большие пакеты, такие, как KDE или GNOME, могут быть установлены в нее, что снимает необходимость расположения их файлов в дереве каталогов /usr. Если Вы собираетесь использовать этот раздел, выделите под него от пяти до десяти гигабайт.
- /tmp – Отдельный раздел для директории /tmp выделяется редко, но будет полезен при настройке тонких клиентов. Данный раздел, если он будет использоваться, не стоит делать больше нескольких гигабайт.
- /usr/src – Этот раздел будет полезно использовать для хранения исходных кодов книги BLFS. Его можно сделать общим между несколькими сборками LFS. Также можно прямо на нем и собирать пакеты BLFS. Раздел размером в 30-50 гигабайт позволит Вам чувствовать себя достаточно свободно.

Любой отдельный раздел, который Вы хотите автоматически подключать при загрузке, необходимо указать в файле /etc/fstab. Подробно о том, как это делать, будет сказано в Раздел 8.2, «Создание файла /etc/fstab».

2.3. Создание файловой системы на разделе

Теперь, когда у нас есть новый чистый раздел, на нем можно создать файловую систему. Самой широкораспространенной в мире Linux файловой системой является вторая расширенная система (ext2, Extended 2 File System), но, с широким распространением жестких дисков большой вместимости, журналируемые файловые системы стремительно набирают популярность. Третья расширенная файловая система (ext3, Extended 3 File System) самая популярная модернизация ext2, которая добавляет возможность журналирования и совместима с утилитами E2fsprogs. Мы создадим файловую систему ext3. Инструкции по созданию других файловых систем можно найти здесь: *http://www.linuxfromscratch.org/blfs/view/svn/postlfs/filesystems.html*.

Чтобы создать файловую систему ext3 на разделе LFS, выполните:

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

Замените <*xxx*> на имя раздела LFS (hda5 в нашем предыдущем примере).

Замечание

Некоторые хост-системы используют собственные расширения в утилитах созданиях файловых систем (E2fsprogs). Это может вызвать проблемы при загрузке в вашу свежую систему LFS в Главе 9, так как эти расширения не будут поддерживаться установленными в LFS E2fsprogs; Вы получите ошибку наподобие «unsupported filesystem features, upgrade your e2fsprogs». Чтобы проверить, использует ли Ваша хост-система собственные расширения, выполните следующую команду:

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

Если вывод содержит другие возможности, кроме has_journal, ext_attr, resize_inode, dir_index, filetype, sparse_super, large_file или needs_recovery, значит Ваш дистрибутив использует собственные расширения. В таком случае, чтобы предотвратить будущие проблемы, Вам стоит скомпилировать пакет E2fsprogs и использовать полученные программы для повторного создания файловой системы на разделе LFS:

```
cd /tmp
tar -xzvf /path/to/sources/e2fsprogs-1.41.14.tar.gz
cd e2fsprogs-1.41.14
mkdir -v build
cd build
../configure
make #note that we intentionally don't 'make install' here!
./misc/mke2fs -jv /dev/<xxx>
cd /tmp
rm -rfv e2fsprogs-1.41.14
```

Если Вы используете уже имеющийся раздел подкачки, нет необходимости форматировать его. Если же Вы создали новый, необходимо его инициализировать следующей командой:

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

Замените *<yyy>* именем раздела подкачки.

2.4. Монтирование нового раздела

Сейчас, когда новый раздел был создан и отформатирован в необходимую файловую систему, необходимо сделать его доступным. Для этого раздел должен быть примонтирован в выбранную точку монтирования. В дальнейшем в этой книге предполагается, что файловая система примонтирована к /mnt/lfs, однако Вы полностью свободны в выборе точки монтирования.

Выберите точку монтирования и присвойте путь до нее переменной LFS командой:

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

Далее, создайте точку монтирования и примонтируйте файловую систему LFS командой:

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

Замените *<xxx>* на имя раздела LFS.

Если Вы используете несколько разделов для LFS (например, один для / и другой для /usr), примонтируйте их с помощью:

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

Замените *<xxx>* и *<yyy>* на правильные имена разделов.

Убедитесь, что новый раздел не подключен с слишком строгими правами (такими как опции nosuid, nodev, или noatime). Запустите **mount** без параметров, чтобы увидеть, какие опции были установлены для раздела LFS. Если nosuid, nodev, и/или noatime установлены, раздел необходимо перемонтировать.

Если Вы используете раздел подкачки, убедитесь, что он включен командой **swapon**:

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

Замените *<zzz>* на имя раздела подкачки.

Теперь, когда подготовлено место для работы, пришло время загрузить пакеты.

Глава 3. Пакеты и патчи

3.1. Вступление

Эта глава содержит список пакетов, которые необходимо загрузить, чтобы построить базовую Linux-систему. Перечисленные версии программного обеспечения проверены и работают, и эта книга основывается на их использовании. Мы категорически не рекомендуем использовать более новые версии, поскольку команды для одной версии могут не работать с более новой. Новейшие версии пакетов также могут содержать ошибки и проблемы, требующие исправления. Эти исправления будут разработаны и стабилизированы в процессе дальнейшей разработки этой книги.

Места размещения пакетов могут время от времени быть недоступны. Если домашняя страница проекта сменилась со времени выпуска этой книги, Google (*http://www.google.com/*) предоставляет полезный поисковый движок, через который Вы найдете большинство (если не все) пакеты. Если поиск не принес успехов, попробуйте альтернативные способы загрузки, обсуждаемые тут: *http://www.linuxfromscratch.org/lfs/packages.html#packages*.

Загруженные пакеты и патчи должны быть сохранены в месте, которое будет доступно на протяжении всего процесса сборки. Также необходима рабочая папка, в которой исходники будут распаковываться и собираться. `$LFS/sources` может быть использована как место для сохранения пакетов и патчей и как рабочая папка. Таким образом, все необходимые элементы будут находиться на разделе LFS и доступны на всех стадиях постороения.

Чтобы создать эту папку, выполните следующую команду от имени `root`, перед тем, как загружать пакеты:

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

Сделайте эту папку доступной для записи и установите бит "клейкости". «Клейкость» означает, что даже если несколько пользователей имеют права на запись в папку, только владелец файла может удалить его из "клейкой" папки. Следующая команда установит биты записи и "клейкости":

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

Простой способ загрузить все пакеты и патчи - использовать файл *wget-list* как входные данные для **wget**. Например:

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

3.2. Все пакеты

Загрузите или получите иным способом нижеследующие пакеты:

• **Autoconf (2.68) - 1,350 KB:**
Домашняя страница: *http://www.gnu.org/software/autoconf/*
Загрузка: *http://ftp.gnu.org/gnu/autoconf/autoconf-2.68.tar.bz2*
MD5 сумма: `864d785215aa60d627c91fcb21b05b07`

• Automake (1.11.1) - 1,042 KB:
Домашняя страница: *http://www.gnu.org/software/automake/*
Загрузка: *http://ftp.gnu.org/gnu/automake/automake-1.11.1.tar.bz2*
MD5 сумма: c2972c4d9b3e29c03d5f2af86249876f

• Bash (4.2) - 6,845 KB:
Домашняя страница: *http://www.gnu.org/software/bash/*
Загрузка: *http://ftp.gnu.org/gnu/bash/bash-4.2.tar.gz*
MD5 сумма: 3fb927c7c33022f1c327f14a81c0d4b0

• Binutils (2.21) - 18,304 KB:
Домашняя страница: *http://www.gnu.org/software/binutils/*
Загрузка: *http://ftp.gnu.org/gnu/binutils/binutils-2.21.tar.bz2*
MD5 сумма: c84c5acc9d266f1a7044b51c85a823f5

• Bison (2.4.3) - 1,614 KB:
Домашняя страница: *http://www.gnu.org/software/bison/*
Загрузка: *http://ftp.gnu.org/gnu/bison/bison-2.4.3.tar.bz2*
MD5 сумма: c1d3ea81bc370dbd43b6f0b2cd21287e

• Bzip2 (1.0.6) - 764 KB:
Домашняя страница: *http://www.bzip.org/*
Загрузка: *http://www.bzip.org/1.0.6/bzip2-1.0.6.tar.gz*
MD5 сумма: 00b516f4704d4a7cb50a1d97e6e8e15b

• Coreutils (8.10) - 11,064 KB:
Домашняя страница: *http://www.gnu.org/software/coreutils/*
Загрузка: *http://ftp.gnu.org/gnu/coreutils/coreutils-8.10.tar.gz*
MD5 сумма: 74d54d09fc5c1bd3337127f49c88b1c5

• DejaGNU (1.4.4) - 1,055 KB:
Домашняя страница: *http://www.gnu.org/software/dejagnu/*
Загрузка: *http://ftp.gnu.org/gnu/dejagnu/dejagnu-1.4.4.tar.gz*
MD5 сумма: 053f18fd5d00873de365413cab17a666

• Diffutils (3.0) - 1,781 KB:
Домашняя страница: *http://www.gnu.org/software/diffutils/*
Загрузка: *http://ftp.gnu.org/gnu/diffutils/diffutils-3.0.tar.gz*
MD5 сумма: 684aaba1baab743a2a90e52162ff07da

• E2fsprogs (1.41.14) - 4,406 KB:
Домашняя страница: *http://e2fsprogs.sourceforge.net/*
Загрузка: *http://prdownloads.sourceforge.net/e2fsprogs/e2fsprogs-1.41.14.tar.gz*
MD5 сумма: 05f70470aea2ef7efbb0845b2b116720

• Expect (5.45) - 614 KB:
Домашняя страница: *http://expect.sourceforge.net/*
Загрузка: *http://prdownloads.sourceforge.net/expect/expect5.45.tar.gz*
MD5 сумма: 44e1a4f4c877e9ddc5a542dfa7ecc92b

• File (5.05) - 583 KB:
Домашняя страница: *http://www.darwinsys.com/file/*
Загрузка: *ftp://ftp.astron.com/pub/file/file-5.05.tar.gz*
MD5 сумма: `0b429063710457be2bd17a18389cb018`

Замечание

File (5.05) может быть недоступен по указанному расположению. Администратор сайта сразу удаляет старые версии пакета, как только выходят новые. Список альтернативных мест скачивания, которые могут иметь необходимую версию пакета, расположен здесь: *http://www.linuxfromscratch.org/lfs/download.html#ftp*.

• Findutils (4.4.2) - 2,100 KB:
Домашняя страница: *http://www.gnu.org/software/findutils/*
Загрузка: *http://ftp.gnu.org/gnu/findutils/findutils-4.4.2.tar.gz*
MD5 сумма: `351cc4adb07d54877fa15f75fb77d39f`

• Flex (2.5.35) - 1,227 KB:
Домашняя страница: *http://flex.sourceforge.net*
Загрузка: *http://prdownloads.sourceforge.net/flex/flex-2.5.35.tar.bz2*
MD5 сумма: `10714e50cea54dc7a227e3eddcd44d57`

• Gawk (3.1.8) - 1,938 KB:
Домашняя страница: *http://www.gnu.org/software/gawk/*
Загрузка: *http://ftp.gnu.org/gnu/gawk/gawk-3.1.8.tar.bz2*
MD5 сумма: `52b41c6c4418b3226dfb8f82076193bb`

• GCC (4.5.2) - 64,774 KB:
Домашняя страница: *http://gcc.gnu.org/*
Загрузка: *http://ftp.gnu.org/gnu/gcc/gcc-4.5.2/gcc-4.5.2.tar.bz2*
MD5 сумма: `d6559145853fbaaa0fd7556ed93bce9a`

• GDBM (1.8.3) - 223 KB:
Домашняя страница: *http://www.gnu.org/software/gdbm/*
Загрузка: *http://ftp.gnu.org/gnu/gdbm/gdbm-1.8.3.tar.gz*
MD5 сумма: `1d1b1d5c0245b1c00aff92da751e9aa1`

• Gettext (0.18.1.1) - 14,785 KB:
Домашняя страница: *http://www.gnu.org/software/gettext/*
Загрузка: *http://ftp.gnu.org/gnu/gettext/gettext-0.18.1.1.tar.gz*
MD5 сумма: `3dd55b952826d2b32f51308f2f91aa89`

• Glibc (2.13) - 15,357 KB:
Домашняя страница: *http://www.gnu.org/software/libc/*
Загрузка: *http://ftp.gnu.org/gnu/glibc/glibc-2.13.tar.bz2*
MD5 сумма: `38808215a7c40aa0bb47a5e6d3d12475`

• GMP (5.0.1) - 1,959 KB:
Домашняя страница: *http://www.gnu.org/software/gmp/*
Загрузка: *http://ftp.gnu.org/gnu/gmp/gmp-5.0.1.tar.bz2*
MD5 сумма: `6bac6df75c192a13419dfd71d19240a7`

• Grep (2.7) - 1,466 KB:
Домашняя страница: *http://www.gnu.org/software/grep/*
Загрузка: *http://ftp.gnu.org/gnu/grep/grep-2.7.tar.gz*
MD5 сумма: e848f07e3e79aa7899345d17c7e4115e

• Groff (1.21) - 3,774 KB:
Домашняя страница: *http://www.gnu.org/software/groff/*
Загрузка: *http://ftp.gnu.org/gnu/groff/groff-1.21.tar.gz*
MD5 сумма: 8b8cd29385b97616a0f0d96d0951c5bf

• GRUB (1.98) - 2,392 KB:
Домашняя страница: *http://www.gnu.org/software/grub/*
Загрузка: *ftp://alpha.gnu.org/gnu/grub/grub-1.98.tar.gz*
MD5 сумма: c0bcf60e524739bb64e3a2d4e3732a59

• Gzip (1.4) - 886 KB:
Домашняя страница: *http://www.gnu.org/software/gzip/*
Загрузка: *http://ftp.gnu.org/gnu/gzip/gzip-1.4.tar.gz*
MD5 сумма: e381b8506210c794278f5527cba0e765

• Iana-Etc (2.30) - 201 KB:
Домашняя страница: *http://freshmeat.net/projects/iana-etc/*
Загрузка: *http://anduin.linuxfromscratch.org/sources/LFS/lfs-packages/conglomeration//iana-etc/iana-etc-2.30.tar.bz2*
MD5 сумма: 3ba3afb1d1b261383d247f46cb135ee8

• Inetutils (1.8) - 1,810 KB:
Домашняя страница: *http://www.gnu.org/software/inetutils/*
Загрузка: *http://ftp.gnu.org/gnu/inetutils/inetutils-1.8.tar.gz*
MD5 сумма: ad8fdcdf1797b9ca258264a6b04e48fd

• IPRoute2 (2.6.37) - 380 KB:
Домашняя страница: *http://www.linuxfoundation.org/collaborate/workgroups/networking/iproute2*
Загрузка: *http://devresources.linuxfoundation.org/dev/iproute2/download/iproute2-2.6.37.tar.bz2*
MD5 сумма: 9774ff9d74ebd301bf56bd8d74473786

• Kbd (1.15.2) - 1,520 KB:
Загрузка: *http://www.kernel.org/pub/linux/utils/kbd/kbd-1.15.2.tar.gz*
MD5 сумма: 77d0b51454522bc6c170bbdc6e31202a

• Less (436) - 297 KB:
Домашняя страница: *http://www.greenwoodsoftware.com/less/*
Загрузка: *http://www.greenwoodsoftware.com/less/less-436.tar.gz*
MD5 сумма: 817bf051953ad2dea825a1cdf460caa4

• LFS-Bootscripts (20100627) - 43 KB:
Загрузка: *http://www.linuxfromscratch.org/lfs/downloads/6.8/lfs-bootscripts-20100627.tar.bz2*
MD5 сумма: 8260bdb271caa3b538f8e95f65998864

• Libtool (2.4) - 2,520 KB:
Домашняя страница: *http://www.gnu.org/software/libtool/*
Загрузка: *http://ftp.gnu.org/gnu/libtool/libtool-2.4.tar.gz*
MD5 сумма: b32b04148ecdd7344abc6fe8bd1bb021

• Linux (2.6.37) - 71,854 KB:
Домашняя страница: *http://www.kernel.org/*
Загрузка: *http://www.kernel.org/pub/linux/kernel/v2.6/linux-2.6.37.tar.bz2*
MD5 сумма: c8ee37b4fdccdb651e0603d35350b434

Замечание

Ядро Linux обновляется очень быстро, зачастую в связи с найденными проблемами безопасности. Стоит использовать последнее доступное ядро версии 2.6.37.x, если страница предупреждений об ошибках не сообщает об ином.

Пользователи с ограниченным трафиком или узким каналом, которые хотят обновить ядро Linux, могут отдельно загрузить базовую версию пакета и патчи к нему. Это может сохранить время или деньги при обновлении ядра до минорного выпуска.

• M4 (1.4.15) - 1,099 KB:
Домашняя страница: *http://www.gnu.org/software/m4/*
Загрузка: *http://ftp.gnu.org/gnu/m4/m4-1.4.15.tar.bz2*
MD5 сумма: c7c32540bc3842d5550f88d47ef551d8

• Make (3.82) - 1,213 KB:
Домашняя страница: *http://www.gnu.org/software/make/*
Загрузка: *http://ftp.gnu.org/gnu/make/make-3.82.tar.bz2*
MD5 сумма: 1a11100f3c63fcf5753818e59d63088f

• Man-DB (2.5.9) - 2,312 KB:
Домашняя страница: *http://www.nongnu.org/man-db/*
Загрузка: *http://download.savannah.gnu.org/releases/man-db/man-db-2.5.9.tar.gz*
MD5 сумма: 9841394f5c5fe7e2dd2e0c5fb4766d0f

• Man-pages (3.32) - 1,108 KB:
Домашняя страница: *http://www.kernel.org/doc/man-pages/*
Загрузка: *http://www.kernel.org/pub/linux/docs/manpages/man-pages-3.32.tar.bz2*
MD5 сумма: 1278c5289660e42a597fefd30d9bdcf0

• Module-Init-Tools (3.12) - 917 KB:
Домашняя страница: *https://modules.wiki.kernel.org/index.php/Module_init_tools_3_12*
Загрузка: *http://www.kernel.org/pub/linux/utils/kernel/module-init-tools/module-init-tools-3.12.tar.bz2*
MD5 сумма: 8b2257ce9abef74c4a44d825d23140f3

• MPC (0.8.2) - 536 KB:
Домашняя страница: *http://www.multiprecision.org/*
Загрузка: *http://www.multiprecision.org/mpc/download/mpc-0.8.2.tar.gz*
MD5 сумма: e98267ebd5648a39f881d66797122fb6

• MPFR (3.0.0) - 1,112 KB:
Домашняя страница: *http://www.mpfr.org/*
Загрузка: *http://www.mpfr.org/mpfr-3.0.0/mpfr-3.0.0.tar.bz2*
MD5 сумма: f45bac3584922c8004a10060ab1a8f9f

• Ncurses (5.7) - 2,388 KB:
Домашняя страница: *http://www.gnu.org/software/ncurses/*
Загрузка: *ftp://ftp.gnu.org/gnu/ncurses/ncurses-5.7.tar.gz*
MD5 сумма: cce05daf61a64501ef6cd8da1f727ec6

• Patch (2.6.1) - 248 KB:
Домашняя страница: *http://savannah.gnu.org/projects/patch/*
Загрузка: *http://ftp.gnu.org/gnu/patch/patch-2.6.1.tar.bz2*
MD5 сумма: `0818d1763ae0c4281bcdc63cdac0b2c0`

• Perl (5.12.3) - 11,759 KB:
Домашняя страница: *http://www.perl.org/*
Загрузка: *http://www.cpan.org/src/5.0/perl-5.12.3.tar.bz2*
MD5 сумма: `72f3f7e1c700e79bbf9d9279ca5b42d9`

• Pkg-config (0.25) - 966 KB:
Домашняя страница: *http://pkg-config.freedesktop.org/*
Загрузка: *http://pkgconfig.freedesktop.org/releases/pkg-config-0.25.tar.gz*
MD5 сумма: `a3270bab3f4b69b7dc6dbdacbcae9745`

• Procps (3.2.8) - 279 KB:
Домашняя страница: *http://procps.sourceforge.net/*
Загрузка: *http://procps.sourceforge.net/procps-3.2.8.tar.gz*
MD5 сумма: `9532714b6846013ca9898984ba4cd7e0`

• Psmisc (22.13) - 373 KB:
Домашняя страница: *http://psmisc.sourceforge.net/*
Загрузка: *http://prdownloads.sourceforge.net/psmisc/psmisc-22.13.tar.gz*
MD5 сумма: `e2c339e6b65b730042084023784a729e`

• Readline (6.2) - 2,225 KB:
Домашняя страница: *http://cnswww.cns.cwru.edu/php/chet/readline/rltop.html*
Загрузка: *http://ftp.gnu.org/gnu/readline/readline-6.2.tar.gz*
MD5 сумма: `67948acb2ca081f23359d0256e9a271c`

• Sed (4.2.1) - 878 KB:
Домашняя страница: *http://www.gnu.org/software/sed/*
Загрузка: *http://ftp.gnu.org/gnu/sed/sed-4.2.1.tar.bz2*
MD5 сумма: `7d310fbd76e01a01115075c1fd3f455a`

• Shadow (4.1.4.3) - 1,762 KB:
Домашняя страница: *http://pkg-shadow.alioth.debian.org/*
Загрузка: *ftp://pkg-shadow.alioth.debian.org/pub/pkg-shadow/shadow-4.1.4.3.tar.bz2*
MD5 сумма: `b8608d8294ac88974f27b20f991c0e79`

• Sysklogd (1.5) - 85 KB:
Домашняя страница: *http://www.infodrom.org/projects/sysklogd/*
Загрузка: *http://www.infodrom.org/projects/sysklogd/download/sysklogd-1.5.tar.gz*
MD5 сумма: `e053094e8103165f98ddafe828f6ae4b`

• Sysvinit (2.88dsf) - 108 KB:
Домашняя страница: *http://savannah.nongnu.org/projects/sysvinit*
Загрузка: *http://download.savannah.gnu.org/releases/sysvinit/sysvinit-2.88dsf.tar.bz2*
MD5 сумма: `6eda8a97b86e0a6f59dabbf25202aa6f`

• Tar (1.25) - 2,273 KB:
Домашняя страница: *http://www.gnu.org/software/tar/*
Загрузка: *http://ftp.gnu.org/gnu/tar/tar-1.25.tar.bz2*
MD5 сумма: `6e497f861c77bbba2f7da4e10270995b`

• Tcl (8.5.9) - 4,365 KB:
Домашняя страница: *http://tcl.sourceforge.net/*
Загрузка: *http://prdownloads.sourceforge.net/tcl/tcl8.5.9-src.tar.gz*
MD5 сумма: 8512d8db3233041dd68a81476906012a

• Texinfo (4.13a) - 2,687 KB:
Домашняя страница: *http://www.gnu.org/software/texinfo/*
Загрузка: *http://ftp.gnu.org/gnu/texinfo/texinfo-4.13a.tar.gz*
MD5 сумма: 71ba711519209b5fb583fed2b3d86fcb

• Udev (166) - 573 KB:
Домашняя страница: *http://www.kernel.org/pub/linux/utils/kernel/hotplug/udev.html*
Загрузка: *http://www.kernel.org/pub/linux/utils/kernel/hotplug/udev-166.tar.bz2*
MD5 сумма: 4db27d73fdbe94f47fd89fdd105c2dfb

• Архив для проверки Udev (166) - 150 KB:
Загрузка: *http://anduin.linuxfromscratch.org/sources/other/udev-166-testfiles.tar.bz2*
MD5 сумма: 64ada14e464dee3388787e3aebf2ac34

• Архив конфигурации Udev - 7 KB:
Загрузка: *http://www.linuxfromscratch.org/lfs/downloads/6.8/udev-config-20100128.tar.bz2*
MD5 сумма: 32de4eb504b2ad67b43cb4fe16da92e2

• Util-linux (2.19) - 4,288 KB:
Домашняя страница: *http://userweb.kernel.org/~kzak/util-linux/*
Загрузка: *http://www.kernel.org/pub/linux/utils/util-linux/v2.19/util-linux-2.19.tar.bz2*
MD5 сумма: 590ca71aad0b254e2631d84401f28255

• Vim (7.3) - 8,675 KB:
Домашняя страница: *http://www.vim.org*
Загрузка: *ftp://ftp.vim.org/pub/vim/unix/vim-7.3.tar.bz2*
MD5 сумма: 5b9510a17074e2b37d8bb38ae09edbf2

• Xz Utils(5.0.1) - 982 KB:
Домашняя страница: *http://tukaani.org/xz*
Загрузка: *http://tukaani.org/xz/xz-5.0.1.tar.bz2*
MD5 сумма: cb6c7a58cec4d663a395c54d186ca0c6

• Zlib (1.2.5) - 532 KB:
Домашняя страница: *http://www.zlib.net/*
Загрузка: *http://www.zlib.net/zlib-1.2.5.tar.bz2*
MD5 сумма: be1e89810e66150f5b0327984d8625a0

Примерный размер всех пакетов: около 286 MB

3.3. Необходимые патчи

В дополнение к пакетам также требуются некоторые патчи. Они исправляют разнообразные ошибки в пакетах, что обычно делают мэйнтейнеры дистрибутивов, и производят небольшие изменения для более простой работы с пакетами. Следующие патчи необходимы для сборки LFS-системы:

• Патч документации Bzip2 - 1.6 KB:
Загрузка: *http://www.linuxfromscratch.org/patches/lfs/6.8/bzip2-1.0.6-install_docs-1.patch*
MD5 сумма: 6a5ac7e89b791aae556de0f745916f7f

• Патч исправления интернационализации Coreutils - 120 KB:
Загрузка: *http://www.linuxfromscratch.org/patches/lfs/6.8/coreutils-8.10-i18n-1.patch*
MD5 сумма: 28895e1112835ca04119158d1883a6d5

• Патч Coreutils Uname - 1.6 KB:
Загрузка: *http://www.linuxfromscratch.org/patches/lfs/6.8/coreutils-8.10-uname-1.patch*
MD5 сумма: 500481b75892e5c07e19e9953a690e54

• Патч кумулятивного обновления Dejagnu - 6 KB:
Загрузка: *http://www.linuxfromscratch.org/patches/lfs/6.8/dejagnu-1.4.4-consolidated-1.patch*
MD5 сумма: c8d481223db274a33b121fb8c25af9f7

• Патч Flex GCC-4.4.x - 1 KB:
Загрузка: *http://www.linuxfromscratch.org/patches/lfs/6.8/flex-2.5.35-gcc44-1.patch*
MD5 сумма: ad9109820534278c6dd0898178c0788f

• Патч GCC Startfiles Fix - 1.5 KB:
Загрузка: *http://www.linuxfromscratch.org/patches/lfs/6.8/gcc-4.5.2-startfiles_fix-1.patch*
MD5 сумма: 799ef1971350d2e3c794f2123f247cc6

• Фикс сборки Glibc с GCC - 2.5 KB:
Загрузка: *http://www.linuxfromscratch.org/patches/lfs/6.8/glibc-2.13-gcc_fix-1.patch*
MD5 сумма: d1f28cb98acb9417fe52596908bbb9fd

• Патч Kbd Backspace/Delete Fix - 12 KB:
Загрузка: *http://www.linuxfromscratch.org/patches/lfs/6.8/kbd-1.15.2-backspace-1.patch*
MD5 сумма: f75cca16a38da6caa7d52151f7136895

• Фикс Patch Testsuite - 1 KB:
Загрузка: *http://www.linuxfromscratch.org/patches/lfs/6.8/patch-2.6.1-test_fix-1.patch*
MD5 сумма: c51e1a95bfc5310635d05081472c3534

• Патч Perl Libc - 1 KB:
Загрузка: *http://www.linuxfromscratch.org/patches/lfs/6.8/perl-5.12.3-libc-1.patch*
MD5 сумма: 800dfd3c9618731ee5cf57f77a7942b4

• Фикс ошибок Procps HZ - 2.3 KB:
Загрузка: *http://www.linuxfromscratch.org/patches/lfs/6.8/procps-3.2.8-fix_HZ_errors-1.patch*
MD5 сумма: 2ea4c8e9a2c2a5a291ec63c92d7c6e3b

• Патч Procps Watch - 3.5 KB:
Загрузка: *http://www.linuxfromscratch.org/patches/lfs/6.8/procps-3.2.8-watch_unicode-1.patch*
MD5 сумма: cd1a757e532d93662a7ed71da80e6b58

Общий размер всех патчей: около 154 KB

В дополнение к данным необходимым патчам, существует множество необязательных патчей, созданных сообществом LFS. Эти патчи решают мелкие проблемы или включают функциональность не предоставляемую по умолчанию. Можете взглянуть на базу патчей *http://www.linuxfromscratch.org/patches/downloads/* и скачать любой патч, который Вам понравится.

Глава 4. Последние приготовления

4.1. О переменной $LFS

На протяжении всей книги будет использоваться переменная окружения LFS. Крайне важно следить за тем, чтобы эта переменная всегда была объявлена. Она должна содержать путь до точки монтирования, выбранной для раздела LFS. Проверьте, что переменная LFS установлена правильно, командой:

```
echo $LFS
```

Убедитесь, что вывод содержит путь до точки монтирования раздела LFS, которой является /mnt/lfs в нашем примере. Если вывод неверен, переменной может быть присвоено правильное значение с помошью нижеследующей команды:

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

Объявление этой переменной необходимо для того, чтобы такие команды, как **mkdir $LFS/tools** можно было ввести как есть или просто скопировать. Оболочка автоматически заменит «$LFS» на «/mnt/lfs» (вместо /mnt/lfs, конечно, будет значение, присвоенное Вами этой переменной) когда будет обрабатывать команду.

Не забывайте проверять, что $LFS объявлена, когда покидаете и вновь входите в рабочее окружение (например, при выполнении команды **su** для получения привелегий root или другого пользователя).

4.2. Создание директории $LFS/tools

Все программы, скомпилированные в Глава 5 будут установлены в директорию $LFS/tools, чтобы отделить их от программ, собираемых в Chapter 6. Эти программы являются временными инструментами и не будут являться частью итоговой системы LFS. Благодаря тому, что они будут установлены в отдельную директорию, их можно будет легко удалить после их использования. Это также не позволит временным программам остаться в рабочих директориях хост-системы в случае ошибки в Глава 5.

Создайте необходимую директорию следующей командой от имени root:

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

Следующим шагом будет создание символической ссылки /tools на хост-системе. Она будет указывать на только что созданную директорию на разделе LFS. Выполните следующую команду (также от имени root):

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

Замечание

Эта команда корректна. Утилита **ln** имеет несколько вариантов указания аргументов, поэтому прочтите **info coreutils ln** и ln(1), прежде чем сообщать нам об ошибке в книге.

Создание символической ссылки позволяет собрать инструментарий так, что он всегда будет использовать абсолютный путь /tools. Это означает, что компилятор, ассемблер и компоновщик будут работать как в главе 5 (где мы все еще используем некоторые инструменты из состава хост-системы), так и в последующих (когда мы с помощью **chroot** переместимся в новое окружение на разделе LFS).

4.3. Добавление пользователя LFS

Если Вы зашли как пользователь root, самая незначительная ошибка может повредить или уничтожить систему. Поэтому мы рекомендуем собирать пакеты в этой главе из-под непривилегированного пользователя. Вы можете использовать своего собственного пользователя, но проще всего установить чистое рабочее окружение, создав нового пользователя lfs, члена новой группы (также именуемой lfs), и исползовать этого пользователя на протяжении всего процесса установки. Выполните следующие команды от имени root , чтобы добавить нового пользователя:

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

Значение опций командной строки:

-s /bin/bash
Это делает **bash** оболочкой по умолчанию для пользователя lfs.

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

-m
Это указывает создать домашнюю директорию для lfs.

-k /dev/null
Этот параметр предотвращает возможное копирование файлов из директории шаблонов (по умолчанию это /etc/skel), изменяя ее местоположение на специальное пустое устройство.

lfs
Это имя для создаваемых пользователя и группы.

Чтобы иметь возможность зайти в систему как пользователь lfs (в противоположность переключению на пользователя lfs когда Вы зашли как root, что не требует наличия пароля у пользователя lfs), присвоим lfs пароль:

```
passwd lfs
```

Дадим lfs полный доступ к директории $LFS/tools, делая пользователя lfs ее владельцем:

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

Если Вы создали отдельную рабочую директорию, как было предложено, необходимо сделать пользователя lfs и ее владельцем также:

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

Далее, зайдите в систему как lfs. Это может быть сделано через виртуальный терминал, менеджер дисплея или с помощью следующей команды:

```
su - lfs
```

Параметр «-» говорит **su** запустить login shell в противоположность non-login shell. Различия между этими двумя типами оболочек подробно изложены в bash(1) и в **info bash**.

4.4. Установка рабочего окружения

Установим правильное рабочее окружение, создав два новых файла настроек для оболочки **bash**. Выполните из-под пользователя lfs следующую команду для создания .bash_profile:

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

Когда Вы зашли под пользовтелем lfs, выполняемая оболочка обычно является так называемой *login shell*, которая считывает файл /etc/profile с хост-системы (обычно содержащий некоторые настройки и переменные окружения, общие для всей системы), а затем файл .bash_profile. Команда **exec env -i../bin/bash** в файле .bash_profile заменяет запущенную оболочку на новую с абсолютно пустым окружением, исключая переменные HOME, TERM и PS1. Это позволяет убедться, что никакие потенциально нежелательные переменные окружения из хост-системы не просочатся в окружение сборки.

Свежезапущенная оболочка представляет собой *non-login shell*, которая не считывает файлы /etc/profile и .bash_profile, вместо этого читая файл .bashrc. Создадим .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 функцию хэширования **bash**. Хэширование чаще всего полезно— **bash** использует хэш-таблицу для запоминания полного пути к исполняемым файлам, чтобы не просматривать заново все каталоги PATH при поиске однажды уже вызванной программы. Но наши инструменты должны включаться в работу сразу после установки. Благодаря отключению функции хеширования оболочка будет всегда просматривать PATH перед выполнением программы, и находить наши свежесобранные инструменты в $LFS/tools как только они станут доступны, не запоминая предыдущие версии тех же программ, расположенные в другом месте.

Устанавливая пользовательскую маску создания файла (umask) в 022, мы указываем, что новые файлы и директории будут доступны для записи только владельцу, но читать и выполнять их смогут все (предполагая значения по умолчанию, исплльзуемые системным вызовом open(2), файлы будут создаваться с правами 644, а папки - 755).

Переменная окружения LFS должна содержать путь до выбранной точки монтирования.

Переменная LC_ALL управляет локализацией некоторых программ, требуя от них форматировать сообщения в соответствии с правилами, принятыми в указанной стране. Если хост-система использует версию Glibc старше 2.2.4, установка переменной LC_ALL в значение, отличное от «POSIX» или «C» (на протяжении этой главы) может привести к неожиданным проблемам, если Вы зайдете во временное окружение с помощью **chroot** и захотите вернуться в него позже. Установка LC_ALL в «POSIX» или «C » (оба значения эквивалентны) гарантирует, что все будет работать так, как ожидается.

Переменная LFS_TGT устанавливает нестандартное, но совместимое определение платформы для использования при сборке наших кросс-компилятора и компоновщика и затем - временного инструментария. Более подробная информация изложена в Раздел 5.2, «Замечания о методе сборки».

Благодаря тому, что мы поместили /tools/bin перед стандартными директориями PATH, все программы, устанавливаемые в Глава 5 будит подхватываться оболочкой сразу после их установки. Это, в совокупности с отключеным хэшированием, минимизирует риск случайного использования старых программ из хост-системы во временном окружении в Главе 5.

Заканчивая подготовку окружения к сборке временных инструментов, считаем только что созданный профиль:

```
source ~/.bash_profile
```

4.5. О SBU

Многие люди хотят знать, хотя бы приблизительно, сколько времени займет компиляция и установка каждого пакета. Поскольку Linux From Scratch может быть собран на многих, абсолютно разных, компьютерах, невозможно привести конкретное время. Самый большой пакет (Glibc) на новейших системах будет собираться около 20 минут, но на старых компьютерах его сборка может затянуться на три дня! Вместо того, чтобы указывать точное время, вводится понятие стандартной единицы сборки (Standard Build Unit, SBU).

Концепция SBU работает следующим образом. Самым первым из пакетов в этой книге компилируется Binutils в Глава 5. Время, которое займет компиляция этого пакета, будет принято за одну стандартную единицу сборки или 1 SBU. Время, требуемое каждому из остальных пакетов на компиляцию, измеряется относительно этого времени.

Например, представим пакет, которому для компиляции требуется 4.5 SBU. Это означает, что если Вам потребовалось 10 минут для компиляции и установки Binutils на первом шаге, сборка этого воображаемого пакета займет *приблизительно* 45 минут. На самом деле, большинство пакетов собираются быстрее, чем Binutils.

В целом, SBU не являются весьма точным способом измерения, поскольку зависят от множества факторов, таких, как версия GCC на хост-системе. Они представлены здесь для того, чтобы дать приблизительное представление о длительности сборки пакета. Поэтому в некоторых случаях время может отличаться в ту или иную сторону на десятки минут.

Чтобы ознакомиться с реальным временем сборки пакетов для некоторых компьютеров, мы рекомендуем посетить домашнюю страницу LinuxFromScratch SBU: *http://www.linuxfromscratch.org/~sbu/*.

Замечание

На большинстве современных систем с несколькими процессорами (или ядрами) время компиляции пакета может быть сокращено за счет выполнения "параллельной сборки". Для этого можно установить соответствующую переменную окружения или непосредственно указать программе **make** количество доступных процессоров. Например, на процессорах Core2Duo можно выбрать компиляцию в 2 потока:

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

или просто выполнять сборку так:

```
make -j2
```

При использовании многопоточной компиляции SBU будут варьироваться гораздо сильнее, чем обычно. Также станет намного сложнее анализировать вывод процесса сборки, поскольку строки от различных потоков перемешаются между собой. Если Вы получили ошибку, Вам прийдется вернуться к однопоточному режиму, чтобы выявить проблему.

4.6. О выполнении тестов

Большинство пакетов предорставляют набор тестов. Запуск тестов для только что собранного пакета - хорошая идея, поскольку это позволит проверить, что все компоненты были скомпилированы корректно. Успешное прохождение пакетом всех тестов обычно гарантирует, что пакет будет работать именно так, как задумано разработчиком. Если тесты провалены, значит в пакете наверняка содержится ошибка.

Некоторые тесты более важны, нежели другие. Например, проверка ключевого набора инструментов—GCC, Binutils и Glibc—является критически необходимой из-за того, что эти пакеты играют главную роль в построении правильно работающей системы. Тесты GCC и Glibc могут занять очень много времени, особенно на старом оборудовании, но настоятельно рекомендуется не пропускать их.

Замечание

Опыт показывает, что немного преждевременно запускать тесты в Глава 5. Дело в том, что хост-система вполне может некоторым образом влиять на них, приводя к неожиданным ошибкам. Поскольку инструменты, собираемые в Глава 5 являются временными и скорее всего будут удалены после сборки системы, мы рекомендуем обычному читателю не выполнять тесты в Глава 5. Инструкции по выполнению этих тестов предоставлены в основном для разработчиков, но и они не обязаны следовать им.

Известная проблема при выполнении тестов Binutils и GCC - исчерпание числа доступных псевдотерминалов (PTY). Из-за этого многие проверки будут провалены. Это может происходить по нескольким причинам, но в самом типичном случае означает, что на хост-дистрибутиве неверно настроена файловая система `devpts`. Эта проблема подробно рассматривается здесь: *http://www.linuxfromscratch.org//lfs/faq.html#no-ptys*.

Иногда проверка пакета завершается неуспешно, но по причинам, которые известны разработчикам и не являются критическими. Сверьтесь с *http://www.linuxfromscratch.org/lfs/build-logs/6.8/* чтобы узнать, известны эти ошибки или нет. Данная страница действительна для всех тестов в этой книге.

Глава 5. Построение временной системы

5.1. Вступление

Эта глава рассказывает, как собрать минимальную Linux-систему. Система будет содержать только инструменты, необходимые для того, чтобы начать построение окончательной LFS-системы в Chapter 6 и предоставляющие несколько более удобное рабочее окружение, чем необходимо для абсолютного минимализма.

Процесс построения этой минимальной системы разбит на две части. Первым шагом будет сборка нового и независимого от хост-системы набора инструментов (компилятора, ассемблера, компоновщика и некоторых полезных утилит). На втором шаге с помощью этого инструментария создаются другие необходимые программы.

Файлы, компилируемые в этой главе, будут устанавливаться в дерево каталогов `$LFS/tools`, чтобы держать их отдельно от файлов, устанавливаемых в следующей главе, и файлов хост-системы. Поскольку собираемые пакеты являются временными, мы не хотим засорять ими будующую LFS-систему.

5.2. Замечания о методе сборки

Эта секция объясняет некоторые моменты и технические детали, относящиеся к методу сборки в целом. Не стоить пытаться понять и запомнить все сразу. Большая часть этой информации будет разъясняться по мере сборки. Можете подглядывать сюда в любое время.

Основная цель Глава 5 - создать временное окружение, содержащее хорошо известный набор инструментов, которое могло бы быть изолировано от хост-системы. Благодаря использованию **chroot**, команды в последующих главах будут выполняться в этом временном окружении, позволяющем собрать LFS чисто и без проблем. Процесс сборки был составлен таким образом, чтобы минимизировать риск для новичков и в то же время предоставить максимум образовательной информации.

Важно

Перед тем, как продолжить сборку, узнайте имя Вашей рабочей платформы. Простейший способ сделать это - запустить скрипт **config.guess**, который поставляется с исходниками многих пакетов. Распакуйте исходные коды Binutils, запустите скрипт: **./config.guess** и запомните выведенное значение. Например, для новейших 32-разрядных процессоров Intel будет выдано что-то вроде *i686-pc-linux-gnu*.

Также узнайте имя динамического компоновщика для Вашей платформы, часто также называемого динамическим загрузчиком (не перепутайте со стандартным компоновщиком **ld**, который является частью Binutils). Динамический компоновщик, предоставляемый Glibc, находит и загружает необходимые для работы программы разделяемые библиотеки, подготавливает программу к запуску и затем запускает ее. Имя динамического компоновщика для 32-разрядной архитектуры Intel - `ld-linux.so.2`. Простой и надежный способ узнать имя динамического компоновщика - проверить любой бинарный исполняемый файл на хост-системе, выполнив: **readelf -l <name of binary> | grep interpreter** и просмотрев вывод. Также надежным источником для любой платформы является файл `shlib-versions` в корне дерева исходников Glibc.

Некоторые ключевые моменты того, как работает метод сборки Глава 5:

- Мы слегка корректируем имя рабочей платформы, изменяя поле "vendor" в переменной `LFS_TGT`, для получения на первом шаге сборки Binutils и GCC совместимых кросс-компилятора и кросс-компоновщика. Вместо того, чтобы создавать бинарные файлы для другой архитектуры, кросс-компилятор и кросс-компоновщик будут собирать программы, совместимые с текущим оборудованием.
- Временные библиотеки собираются при помощи кросс-компиляции. Поскольку кросс-компилятор по своей сути не может полагаться на что-либо из своей хост-системы, этот метод снимает возможность потенциального загрязнения целевой системы, снижая шанс объединения заголовочных файлов или библиотек хост-системы с новыми инструментами. Кросс-компиляция также дает возможность собрать и 32-разрядные, и 64-разрядные библиотеки на 64-разрядном оборудовании.
- Аккуратная манипуляция с файлом спецификаций **gcc** позволяет указать компилятору, какой динамический компоновщик следует использовать.

Binutils устанавливается первым, поскольку скрипты **configure** и в GCC, и в Glibc, проводят тестирование возможностей имеющихся ассемблера и компоновщика для определения программных компонентов, которые им необходимо задействовать или отключить. Это намного важнее, чем кажется на первый взгляд. Неверно настроенные GCC и Glibc могут привести к появлению в инструментарии очень хитрых ошибок, которые могут не проявлять себя почти до окончания сборки всего дистрибутива. Выполнение тестирований обычно позволяет обнаружить такие ошибки, прежде чем будет построена большая часть системы.

Binutils устанавливает свои ассемблер и компоновщик в две директории, `/tools/bin` и `/tools/$LFS_TGT/bin`. Утилиты в одной из папок являются жесткими ссылками на утилиты в другой. Важным аспектом конфигурации компоновщика

является путь поиска библиоетк, используемый им по умолчанию. Подробную информацию можно получить, запустив **ld** с флагом *--verbose*. Например, команда **ld --verbose | grep SEARCH** покажет текущие пути поиска и порядок, в котором компоновщик просматривает их. Можно просмотреть, какие файлы компонуются с пустой программой, компилируя ее и передавая компоновщику ключ *--verbose*. Например, **gcc dummy.c -Wl,--verbose 2>&1 | grep succeeded** покажет фсе файлы, успешно подключенные к программе на стадии компоновки.

Следующим по порядку пакетом устанавливается GCC. Пример части вывода его скрипта **configure** приведен ниже:

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

This is important for the reasons mentioned above. It also demonstrates that GCC's configure script does not search the PATH directories to find which tools to use. However, during the actual operation of **gcc** itself, the same search paths are not necessarily used. To find out which standard linker **gcc** will use, run: **gcc -print-prog-name=ld**.

Detailed information can be obtained from **gcc** by passing it the *-v* command line option while compiling a dummy program. For example, **gcc -v dummy.c** will show detailed information about the preprocessor, compilation, and assembly stages, including **gcc**'s included search paths and their order.

The next package installed is Glibc. The most important considerations for building Glibc are the compiler, binary tools, and kernel headers. The compiler is generally not an issue since Glibc will always use the compiler relating to the *--host* parameter passed to its configure script, e.g. in our case, **i686-lfs-linux-gnu-gcc**. The binary tools and kernel headers can be a bit more complicated. Therefore, take no risks and use the available configure switches to enforce the correct selections. After the run of **configure**, check the contents of the `config.make` file in the `glibc-build` directory for all important details. Note the use of *CC="i686-lfs-gnu-gcc"* to control which binary tools are used and the use of the *-nostdinc* and *-isystem* flags to control the compiler's include search path. These items highlight an important aspect of the Glibc package—it is very self-sufficient in terms of its build machinery and generally does not rely on toolchain defaults.

After the Glibc installation, change **gcc**'s specs file to point to the new dynamic linker in `/tools/lib`. This last step is vital in ensuring that searching and linking take place only within the `/tools` prefix. A hard-wired path to a dynamic linker is embedded into every Executable and Link Format (ELF)-shared executable. This can be inspected by running: **readelf -l <name of binary> | grep interpreter**. Amending **gcc**'s specs file ensures that every program compiled from here through the end of this chapter will use the new dynamic linker in `/tools/lib`.

For the second pass of GCC, its sources also need to be modified to tell GCC to use the new dynamic linker. Failure to do so will result in the GCC programs themselves having the name of the dynamic linker from the host system's `/lib` directory embedded into them, which would defeat the goal of getting away from the host.

During the second pass of Binutils, we are able to utilize the *--with-lib-path* configure switch to control **ld**'s library search path. From this point onwards, the core toolchain is self-contained and self-hosted. The remainder of the Глава 5 packages all build against the new Glibc in `/tools`.

Upon entering the chroot environment in Chapter 6, the first major package to be installed is Glibc, due to its self-sufficient nature mentioned above. Once this Glibc is installed into `/usr`, we will perform a quick changeover of the toolchain defaults, and then proceed in building the rest of the target LFS system.

5.3. Общие инструкции по сборке

При сборке пакетов необходимо иметь в виду следующее:

- К некоторым пакетам перед компиляцией применяются патчи, но только если они необходимы, чтобы обойти какую-либо проблему. Чаще всего патч будет необходим в обеих главах (этой и следующей), но иногда только в одной или другой. Поэтому, не волнуйтесь, если Вам кажется, что инструкции по применению патча пропущены. Также не стоит беспокоиться о сообщениях, сообщающих об *offset* (смещении) или *fuzz* при применении патча. Патч все равно был применен успешно.
- Во время компиляции пакета на экране могут появляться предупреждения. Это нормально, не обращайте на них внимания. Скорее всего, эти предупреждения говорят об использовании устаревшего, но не являющимся неверным синтаксиса C/C++. Стандарты на язык C меняются достаточно часто, и некоторые пакеты используют их старые версии. Это не проблема, просто предупреждение.

Важно

После установки каждого пакета удалите директории исходников и сборки, если нет иных предписаний. Удаление исходников предотвращает неверное конфигурирование при последующей повторной установке пакета.

- Последний раз проверьте, что переменная окружения LFS объявлена верно:

```
echo $LFS
```

Убедитесь, что в выводе присутствует верный путь до точки монтирования раздела LFS, которым в нашем примере является `/mnt/lfs`.

- Прежде чем приступить к сборке, нужно подчеркнуть следующее:

Важно

Инструкции по сборке написаны для командной оболочки **bash**

Важно

Перед выполнением инструкций по сборке пакета необходимо распаковать его от имени пользователя `lfs` и перейти в созданную директорию командой **cd**.

Еще раз напомним процесс сборки:

```
1. Поместите все исходные коды и патчи в директорию, которая будет
доступна из временного окружения, например, /mnt/lfs/sources/.
Не кладите исходники в /mnt/lfs/tools/!
2. Смените текущую директорию на директорию с исходниками.
3. Для каждого пакета:
a. Используя утилиту tar, извлеките архив.
b. Переместитесь в каталог, созданный при распаковке пакета.
c. Следуйте инструкциям по сборке пакета, приведенным здесь.
d. Выйдите из каталога с исходными кодами пакета.
e. Удалите директорию с исходниками и все директории <имяпакета>-build,
если они были созданы в процессе сборки.
```

5.4. Binutils-2.21 - Шаг 1

The Binutils package contains a linker, an assembler, and other tools for handling object files.

Приблизительное время сборки:	1 SBU
Требует свободного места на диске:	248 MB

5.4.1. Установка Cross Binutils

Замечание

Вернитесь и внимательно прочитайте замечания в предыдущей секции. Постарайтесь их запомнить, это поможет Вам избежать многих проблем позже.

Критически важно, чтобы пакет Binutils компилировался самым превым. И Glibc, и GCC выполняют различные проверки компоновщика и ассемблера для того, чтобы определить возможости, которые они могут задействовать.

Документация Binutils рекомендует собирать Binutils вне директории, содержащей исходные коды. Создадим отдельную директорию:

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

Замечание

Чтобы использовать систему SBU, нужно засечь время, необходимое для конфигурации, сборки и установки пакета Binutils. Проще всего это сделать, обернув все три команды с помощью утилиты **time** примерно так: **time { ./configure ... && make && make install; }**.

Замечание

Приближенные значения SBU и указанное необходимое место на диске в главе 5 берутся без учета выполнения тестов.

Подготовим Binutils к компиляции:

```
../binutils-2.21/configure \
    --target=$LFS_TGT --prefix=/tools \
    --disable-nls --disable-werror
```

Значение ключей configure:

`--target=$LFS_TGT`
Поскольку описание системы в переменной `LFS_TGT` немного отличается от значения, которое вернет скрипт **config.guess**, этот ключ укажет скрипту **configure** настроить систему сборки Binutils для компиляции кросс-компоновщика.

--prefix=/tools
Этот ключ указывает скрипту configure подготовить утилиты Binutils для установки в директорию /tools.

--disable-nls
Это отключает интернационализацию, поскольку она не нужна для временных инструментов.

--disable-werror
Эта опция отключает интерпретацию предупреждений хост-компилятора как ошибок. Предупреждения компилятора далеко не всегда означают наличие каких-то проблем, см Раздел 5.3, «Общие инструкции по сборке.».

Скомпилируем пакет:

```
make
```

Компиляция завершена. Обычно мы бы запустили тестирование пакета, но на этой ранней стадии пакеты, необходимые для него (Tcl, Expect и DejaGNU), еще не установлены. Выгода от запуска тестирования на данном этапе минимальна, поскольку программы, собранные на первом шаге, вскоре будут заменены программами, которые мы соберем на втором.

Если сборка производится на архитектуре x86_64, создайте необходимую символическую ссылку:

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

Установим пакет:

```
make install
```

Подробная информация об этом пакете расположена в Раздел 6.12.2, «Contents of Binutils.»

5.5. GCC-4.5.2 - Шаг 1

The GCC package contains the GNU compiler collection, which includes the C and C++ compilers.

Приблизительное время сборки:	5.0 SBU
Требует свободного места на диске:	809 MB

5.5.1. Установка Кросс-GCC

Для GCC необходимы пакеты GMP, MPC и MPFR. Поскольку они могут быть не включены в Ваш хост-дистрибутив, необходимо собрать их вместе с GCC. Распакуем каждый пакет в директорию, содержащую исходный код GCC, и переименуем получившиеся подкаталоги таким образом, чтобы скрипты сборки GCC смогли найти их и автоматически задействовать:

```
tar -jxf ../mpfr-3.0.0.tar.bz2
mv -v mpfr-3.0.0 mpfr
tar -jxf ../gmp-5.0.1.tar.bz2
mv -v gmp-5.0.1 gmp
tar -zxf ../mpc-0.8.2.tar.gz
mv -v mpc-0.8.2 mpc
```

Документация GCC рекомендует собирать GCC вне директории, содержащей исходники. Создадим отдельную директорию:

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

Подготовим GCC к компиляции:

```
../gcc-4.5.2/configure \
    --target=$LFS_TGT --prefix=/tools \
    --disable-nls --disable-shared --disable-multilib \
    --disable-decimal-float --disable-threads \
    --disable-libmudflap --disable-libssp \
    --disable-libgomp --enable-languages=c \
    --with-gmp-include=$(pwd)/gmp --with-gmp-lib=$(pwd)/gmp/.libs \
    --without-ppl --without-cloog
```

Значение опций configure:

`--disable-shared`
Этот ключ заставляет GCC компоноваться со своими внутренними библиотеками статически. Мы делаем это, чтобы предотвратить возможное влияние со стороны хост-системы.

`--disable-decimal-float, --disable-threads, --disable-libmudflap, --disable-libssp, --disable-libgomp`
Эти опции убирают поддержку плавающей точки, многопоточности, библиотек libmudflap, libssp и libgomp соответственно. Если не отключить эти расширения,

сборка кросс-компилятора завершится неудачей. Кроме того, эти возможности не являются необходимыми для кросс-компиляции временной библиотеки C.

--disable-multilib
На x86_64 LFS еще не поддерживает мультиархитектурную конфигурацию. Для x86 этот ключ игнорируется.

--enable-languages=c
Этот ключ указывает, что необходимо собрать только компилятор C, поскольку другие языки нам сейчас не нужны.

--with-gmp-include=...
Подсказывает GCC местонахождение заголовочных файлов GMP.

--with-gmp-lib=...
Указывает GCC, где расположена библиотека GMP.

--without-ppl, --without-cloog
Эти ключи предотвращают сборку GCC с поддержкой библиотек PPL и CLooG, которые могут присутствовать на хост-системе, но будут недоступны в среде chroot.

Скомпилируем GCC, запустив:

```
make
```

Компиляция завершена. На данном этапе мы можем запустить тестирование пакета, хотя и еще не все необходимые для этого инструменты установлены. Однако, выгоды от запуска тестирования сейчас минимальны, поскольку программы, собранные на этом, первом, шаге, вскоре будут заменены.

Установим пакет:

```
make install
```

Поскольку мы использовали параметр *--disable-shared*, файл `libgcc_eh.a` не был создан и установлен. Пакет Glibc зависит от этого файла, поскольку использует *-lgcc_eh* в своей системе сборки. Можно исправить ситуацию, создав символьную ссылку на `libgcc.a`, так как этот файл содержит объекты, обычно находящиеся в `libgcc_eh.a`:

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

Подробная информация об этом пакете расположена в Раздел 6.16.2, «Contents of GCC.»

5.6. Linux-2.6.37 API Headers

The Linux API Headers expose the kernel's API for use by Glibc.

Приблизительное время сборки:	0.1 SBU
Требует свободного места на диске:	485 MB

5.6.1. Установка Linux API Headers

Необходимо определить программный интерфейс приложения (Application Programming Interface, API) ядра Linux, который будет использовать системная библиотека C (Glibc в LFS). Для этого необходимо установить различные заголовочные файлы, поставляемые с ядром Linux.

Убедимся, что не осталось никаких старых файлов и зависимостей от предыдущих действий:

```
make mrproper
```

Теперь проверим и извлечем заголовочные файлы пользовательского уровня из исходных кодов. Они помещаются во временную директорию и только потом копируются в необходимое месторасположение потому, что процесс извлечения удаляет все в директории назначения.

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

Подробная информация об этом пакете расположена в Раздел 6.7.2, «Contents of Linux API Headers.»

5.7. Glibc-2.13

The Glibc package contains the main C library. This library provides the basic routines for allocating memory, searching directories, opening and closing files, reading and writing files, string handling, pattern matching, arithmetic, and so on.

Приблизительное время сборки:	6.9 SBU
Требует свободного места на диске:	371 MB

5.7.1. Установка Glibc

Исправим ошибку, из-за которой не удается собрать Glibc при помощи GCC-4.5.2:

```
patch -Np1 -i ../glibc-2.13-gcc_fix-1.patch
```

Документация Glibc рекомендует собирать Glibc вне дерева исходников. Создадим отдельную директорию:

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

Поскольку Glibc больше не поддерживает i386, ее разработчики рекомендуют использовать флаг *-march=i486* при сборке на архитектуре x86. Это можно сделать несколькими способами, но опыт показывает, что этот флаг лучше всего поместить в переменную окружения «CFLAGS». Вместо того, чтобы полностью переопределять эту переменную, добавим новый флаг к уже существующему содержимому CFLAGS, используя специальный файл `configparms`. Флаг -mtune=native также необходим для присвоения -mtune верного значения при изменении флага -march.

```
case `uname -m` in
  i?86) echo "CFLAGS += -march=i486 -mtune=native" > configparms ;;
esac
```

Подготовим Glibc к компиляции:

```
../glibc-2.13/configure --prefix=/tools \
    --host=$LFS_TGT --build=$(../glibc-2.13/scripts/config.guess) \
    --disable-profile --enable-add-ons \
    --enable-kernel=2.6.22.5 --with-headers=/tools/include \
    libc_cv_forced_unwind=yes libc_cv_c_cleanup=yes
```

Значение опций configure:

--host=$LFS_TGT, --build=$(../glibc-2.13/scripts/config.guess)
Комбинация этих ключей укажет системе сборки Glibc настроить пакет для кросс-компиляции с использованием кросс-компоновщика и кросс-компилятора из директории `/tools`.

--disable-profile
Это запрещает включение информации для профилирования в библиотеки. Опустите эту опцию, если профилирование необходимо для временных инструментов.

--enable-add-ons
Говорит Glibc использовать дополнение NPTL как библиотеку потоков.

--enable-kernel=2.6.22.5
Указывает Glibc собрать библиотеки с поддержкой ядер Linux 2.6.22.5 и новее. Код, необходимый для работы с более старыми ядрами, не будет включен.

--with-headers=/tools/include
Говорит Glibc компилировать себя с использованием заголовков, установленных в директорию инструментария, благодаря чему библиотека будет знать возможности ядра и может соответствующим образом оптимизировать себя.

libc_cv_forced_unwind=yes
Компоновщик, установленный в Раздел 5.4, «Binutils-2.21 - Шаг 1» был собран методом кросс-компиляции и поэтому не может быть использован до установки Glibc. Это означает, что проверка configure на поддержку компоновщиком force-unwind закончится неудачно. Переменная libc_cv_forced_unwind=yes информирует **configure**, что поддержка force-unwind доступна и тестирование этой возможности можно пропустить.

libc_cv_c_cleanup=yes
Аналогично, мы передаем libc_cv_c_cleanup=yes скрипту **configure** для пропуска тестирования заведомо работающей поддержки C cleanup handling support.

На этой стадии вы можете увидеть следующее сообщение:

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

Отсутствующая или несовместимая утилита **msgfmt** обычно не приводит к проблемам. Эта программа является частью пакета Gettext, который должен предоставляться хост-системой.

Скомпилируем пакет:

```
make
```

Этот пакет поставляется с набором тестов, однако, мы не сможем их запустить сейчас, поскольку еще не установили компилятор C++.

Замечание

Для успешного выполнения тестирования также необходимо установить данные локалей. Данные локалей предоставляют системе информацию о том, в каком формате необходимо осуществлять ввод и вывод даты, времени и валют. Если тестирование (как и рекомендуется) не будет запускаться в этой главе, нет нужды устанавливать локали сейчас. Необходимые локали будут установлены в следующей главе. Чтобы все же установить локали Glibc, следуйте инструкциям из Раздел 6.9, «Glibc-2.13.»

Установим пакет:

```
make install
```

Подробная информация об этом пакете расположена в Раздел 6.9.4, «Contents of Glibc.»

5.8. Корректировка инструментария

Теперь, когда временные библиотеки C установлены, все инструменты, скомпилированные в оставшейся части главы, должны быть скомпонованы с этими библиотеками. Для этого необходимо изменить файл спецификаций кросс-компилятора, чтобы он указывал на новый динамический компоновщик в `/tools`.

Это производится копированием файла «спецификаций» компилятора в место, где он ищет этот файл по умолчанию. Затем с помощью **sed** заменяется динамический компоновщик, который будет использоваться GCC по умолчанию. Необходимо найти все ссылки на динамический компоновщик, расположенный в `/lib` или `/lib64`, если хост-система 64-разрядна, и заменить их так, чтобы они указывали на наш новый компоновщик в `/tools`.

Для точности рекомендуется использовать метод копирования-вставки при выполнении следующей команды. Тщательно проверьте файл спецификаций визуально, чтобы убедиться, что она верно заменила все ссылки на расположение динамического комполовщика. Если необходимо, обратитесь к Раздел 5.2, «Замечания о методе сборки,» за правильным именем компоновщика.

```
SPECS=`dirname $($LFS_TGT-gcc -print-libgcc-file-name)`/specs
$LFS_TGT-gcc -dumpspecs | sed \
  -e 's@/lib\(64\)\?/ld@/tools&@g' \
  -e "/^\*cpp:$/{n;s,$, -isystem /tools/include,}" > $SPECS
echo "New specs file is: $SPECS"
unset SPECS
```

Предостережение

На данном этапе критически важно остановиться и проверить, что базовые операции (компиляция и связывание) нового инструментария работают корректно. Для этого выполните следующие команды:

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

Если все работает верно, вывод последней команды должен содержать примерно следующее:

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

Помните, что префиксом динамического компоновщика является `/tools/lib` или `/tools/lib64` для 64-разрядных систем.

Если вывод не совпадает с вышеприведенным, или вообще ничего не выводится, что-то работает неверно. Проверьте и обдумайте все шаги, чтобы выяснить, где Вы допустили неточность, и исправьте ее. Подобную проблему необходимо решить перед тем, как продолжать сборку. Возможно, произошла ошибка при изменении файла спецификаций. В этом случае повторите команду, точно скопировав ее из книги.

Если все прошло успешно, удалим тестовые файлы:

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

Замечание

Сборка Binutils в следующей секции предоставит дополнительный шанс проверить, что инструментарий работает корректно. Если сборка Binutils завершается с ошибкой, это говорит о том, что что-то пошло не так с установкой предыдущих Binutils, GCC или Glibc.

5.9. Binutils-2.21 - Шаг 2

The Binutils package contains a linker, an assembler, and other tools for handling object files.

Приблизительное время сборки:	1.3 SBU
Требует свободного места на диске:	259 MB

5.9.1. Установка Binutils

Снова создадим отдельную директорию для сборки:

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

Подготовим Binutils к компиляции:

```
CC="$LFS_TGT-gcc -B/tools/lib/" \
   AR=$LFS_TGT-ar RANLIB=$LFS_TGT-ranlib \
   ../binutils-2.21/configure --prefix=/tools \
   --disable-nls --with-lib-path=/tools/lib
```

Значение новых ключей configure:

`CC="$LFS_TGT-gcc -B/tools/lib/" AR=$LFS_TGT-ar RANLIB=$LFS_TGT-ranlib`
Установка этих переменных необходима для того, чтобы при сборке использовались наши инструменты кросс-компиляции, а не утилиты хост-системы.

`--with-lib-path=/tools/lib`
Это указывает скрипту configure задать для компиляции наш собственный путь поиска библиотек. В результате `/tools/lib` будет передано компоновщику, что предотвратит поиск библиотек в папках хост-системы.

Скомпилируем пакет:

```
make
```

Установим пакет:

```
make install
```

Подготовим компоновщик к фазе «корректировки» в следующей главе:

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

Значение параметров make:

`-C ld clean`
Это указывает утилите make удалить все скомпилированные программы в поддиректории `ld` .

`-C ld LIB_PATH=/usr/lib:/lib`
Эта опция требует пересобрать все в поддиректории `ld`. Указание `LIB_PATH` позволяет переопределить одноименную переменную в Makefile, указывающую на наши временные инструменты, и заставить компоновщик использовать правильное значение, необходимое для конечной системы. Эта переменная определяет путь по умолчанию, который использует компоновщик для поиска библиотек.

Подробная информация об этом пакете расположена в Раздел 6.12.2, «Contents of Binutils.»

5.10. GCC-4.5.2 - Шаг 2

The GCC package contains the GNU compiler collection, which includes the C and C++ compilers.

Приблизительное время сборки:	9.0 SBU
Требует свободного места на диске:	1003 MB

5.10.1. Установка GCC

Версии GCC новее, чем 4.3, ошибочно попытаются собрать перемещаемый компилятор, который не будет просматривать директорию, указанную с помощью ключа *--prefix*, в поисках стандартных системных библиотек. Поскольку нам нужен вовсе не перемещаемый компилятор, и стандартные системные библиотеки находятся в директории `/tools`, критически необходимо собрать компилятор, скомпонованный с ними. Применим следующий патч, частично возвращающий GCC его старое поведение:

```
patch -Np1 -i ../gcc-4.5.2-startfiles_fix-1.patch
```

В обычных обстоятельствах GCC запускает скрипт **fixincludes** , чтобы поправить потенциально испорченные заголовочные файлы. Поскольку GCC-4.5.2 и Glibc-2.13 к этому времени уже установлены, и с их заголовочными файлами все в порядке, нет необходимости запускать скрипт **fixincludes**. На самом деле этот скрипт может испортить окружение сборки, установив "исправленные" заголовочные файлы из хост-системы во внутреннюю директорию включаемых файлов GCC. Можно отключить запуск скрипта **fixincludes**, выполнив следующие команды:

```
cp -v gcc/Makefile.in{,.orig}
sed 's@\./fixinc\.sh@-c true@' gcc/Makefile.in.orig > gcc/Makefile.in
```

На архитектуре x86, цепная сборка GCC использует флаг компилятора `-fomit-frame-pointer`. Нецепная сборка по умолчанию опускает его, и нашей целью является компилятор, который был бы таким же, как будто он был собран при помощи цепной компиляции. Примените следующую команду **sed**, чтобы заставить систему сборки GCC использовать этот флаг всегда:

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

Следующая команда изменит компоновщик, используемый GCC по умолчанию, на установленный нами в `/tools`. Также она удалит `/usr/include` из путей поиска включаемых файлов GCC. Выполнение этих действий сейчас вместо исправления

файла спецификаций компилятора после установки позволяет использовать новый динамический компоновщик уже при сборке GCC. Благодаря этому все программы, созданные при сборке, будут скомпонованы с новой Glibc. Выполните:

```
for file in \
 $(find gcc/config -name linux64.h -o -name linux.h -o -name sysv4.h)
do
  cp -uv $file{,.orig}
  sed -e 's@/lib\(64\)\?\(32\)\?/ld@/tools&@g' \
  -e 's@/usr@/tools@g' $file.orig > $file
  echo '
#undef STANDARD_INCLUDE_DIR
#define STANDARD_INCLUDE_DIR 0
#define STANDARD_STARTFILE_PREFIX_1 ""
#define STANDARD_STARTFILE_PREFIX_2 ""' >> $file
  touch $file.orig
done
```

Если строки выше кажутся Вам непонятными, давайте притормозим и разберемся. Сначала мы находим в директории `gcc/config` и ее поддиректориях все файлы с именами `linux.h`, `linux64.h` или `sysv4.h`. Каждый найденный файл мы копируем в файл с тем же именем, добавляя к нему суффикс «.orig». Затем первое выражение sed подставляет «/tools» в начало каждого вхождения «/lib/ld», «/lib64/ld» или «/lib32/ld », а второе заменяет жестко зашитые вхождения «/usr». После этого мы добавляем наши определения, которые изменяют пути поиска заголовочных файлов и префикс библиотек по умолчанию, в конец файла. В самом конце мы используем **touch** для обновления временных меток скопированных файлов. Благодаря этому трюку и использованию **cp -u**, мы предотвращаем ошибочные изменения оригинальных файлов в случае нечаянного повторного выполнения команд.

На x86_64 отключим опцию мультиархитектурности для GCC, что не позволит ему скомпоноваться с 32-разрядными библиотеками, которые могут присутствовать на хост-системе:

```
case $(uname -m) in
  x86_64)
    for file in $(find gcc/config -name t-linux64) ; do \
      cp -v $file{,.orig}
      sed '/MULTILIB_OSDIRNAMES/d' $file.orig > $file
    done
  ;;
esac
```

Как и при первой сборке GCC, необходимы пакеты GMP, MPC и MPFR. Распакуем архивы исходников и присвоим директориям требуемые имена:

```
tar -jxf ../mpfr-3.0.0.tar.bz2
mv -v mpfr-3.0.0 mpfr
tar -jxf ../gmp-5.0.1.tar.bz2
mv -v gmp-5.0.1 gmp
tar -zxf ../mpc-0.8.2.tar.gz
mv -v mpc-0.8.2 mpc
```

Снова создадим отдельную директорию для сборки:

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

Перед тем, как приступить к сборке GCC, не забудьте сбросить все переменные окружения, переопределяющие флаги оптимизации.

Теперь подготовим GCC к компиляции:

```
CC="$LFS_TGT-gcc -B/tools/lib/" \
    AR=$LFS_TGT-ar RANLIB=$LFS_TGT-ranlib \
    ../gcc-4.5.2/configure --prefix=/tools \
    --with-local-prefix=/tools --enable-clocale=gnu \
    --enable-shared --enable-threads=posix \
    --enable-__cxa_atexit --enable-languages=c,c++ \
    --disable-libstdcxx-pch --disable-multilib \
    --disable-bootstrap --disable-libgomp \
    --with-gmp-include=$(pwd)/gmp --with-gmp-lib=$(pwd)/gmp/.libs \
    --without-ppl --without-cloog
```

Значение новых опций configure:

--enable-clocale=gnu
Благодаря этой опции в любой ситуации будет выбрана правильная модель локали для библиотек C++. Если скрипт configure найдет установленную локаль *de_DE*, он выберет корректную модель локали gnu. Однако, если локаль *de_DE* не установлена, есть риск того, что будут собраны ABI (Application Binary Interface) -несовместимые библиотеки C++, поскольку может быть выбрана неверная модель локали.

--enable-threads=posix
Включает обработку исключений C++ для многопоточных приложений.

--enable-__cxa_atexit
Разрешает использование `__cxa_atexit` вместо `atexit` для регистрации деструкторов C++ локальных статических и глобальных объектов. Эта опция необходима для полностью совместимой со стандартами обработки деструкторов. Также она затрагивает C++ ABI, поэтому компилируемые в дальнейшем разделяемые библиотеки и программы C++ могут без проблем использоваться в другом дистрибутиве Linux.

--enable-languages=c,c++
Эта опция указывает собрать только компиляторы C и C++.

--disable-libstdcxx-pch
Запрещает сборку прекомпилированного заголовка (precompilied header, PCH) для `libstdc++`. Он займет много места и мы все равно не будем его использовать.

--disable-bootstrap
По умолчанию GCC использует "цепную" систему сборки. При этом GCC компилируется несколько раз. Программы, компилируемые на первом шаге, используются для компиляции самих себя повторно, и затем цикл повторяется еще раз. Программы, собранные на второй и третьей стадиях, сравниваются, чтобы удостовериться, что они могут воспроизводить сами себя без изъянов. Также это позволяет проверить корректность компиляции. Метод сборки,

применяемый в LFS, предоставляет чистый компилятор без необходимости цепной сборки, поэтому мы ее отключаем.

Скомпилируем пакет:

```
make
```

Установим пакет:

```
make install
```

Как последний штрих, создадим символическую ссылку. Многие программы и скрипты запускают **cc** вместо **gcc**, что сохраняет пакеты портируемыми на любую UNIX-систему, где не обязательно установлен GNU C Compiler. Запуск **cc** позволяет системному администратору свободно выбирать используемый компилятор C:

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

Предостережение

На данном этапе критически важно остановиться и проверить, что базовые операции (компиляция и связывание) нового инструментария работают корректно. Для этого выполните следующие команды:

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

Если все работает верно, вывод последней команды должен содержать примерно следующее:

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

Помните, что префиксом динамического компоновщика является `/tools/lib` или `/tools/lib64` для 64-разрядных систем.

Если вывод не совпадает с вышеприведенным, или вообще ничего не выводится, что-то работает неверно. Проверьте и обдумайте все шаги, чтобы выяснить, где Вы допустили неточность, и исправьте ее. Подобную проблему необходимо решить перед тем, как продолжать сборку. В первую очередь выполните проверку заново, используя **gcc** вместо **cc**. Если это работает, значит проблемы с символической ссылкой `/tools/bin/cc`. Пересоздайте ссылку так, как указано выше. Если же это не помогло, убедитесь, что переменная `PATH` содержит верное значение. Это можно сделать, выполнив **echo $PATH** и проверив, что директория `/tools/bin` находится в начале списка. Если это не так, возможно, Вы зашли не как пользователь `lfs` или что-то было сделано неверно в Раздел 4.4, «Установка рабочего окружения.»

Если все прошло успешно, удалим тестовые файлы:

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

Подробная информация об этом пакете расположена в Раздел 6.16.2, «Contents of GCC.»

5.11. Tcl-8.5.9

Пакет Tcl содержит Tool Command Language, управляющий язык инструментов.

Приблизительное время сборки:	0.5 SBU
Требует свободного места на диске:	32 MB

5.11.1. Установка Tcl

Этот пакет и два следующих (Expect и DejaGNU) устанавливаются для того, чтобы иметь возможность запустить тестирование GCC и Binutils. Установка трех пакетов только для этих целей пожет показаться излишней, но очень важно, если не необходимо, проверить, что самые главные инструменты работают верно. Даже если Вы не планируете запускать тесты в этой главе (они не являются обязательными), эти пакеты будут необходимы в Chapter 6.

Подготовим Tcl к компиляции:

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

Скомпилируем пакет:

```
make
```

Компиляция завершена. Как говорилось ранее, необязательно выполнять тесты для временных инструментов в этой главе. Чтобы все же запустить тестирование Tcl, выполните следующую команду:

```
TZ=UTC make test
```

Тестирование Tcl может завершиться неудачно при некоторых условиях хост-системы, которые не до конца известны. Поэтому неудачное окончание проверок здесь не является сюрпризом, и не следует его считать критичным. Параметр *TZ=UTC* устанавливает временную зону в универсальное скоординированное время (UTC), также известное как GMT (Greenwich Mean Time), но только на период тестирования. Благодаря этому проверки часов должны пройти успешно. Подробнее о переменной окружения TZ будет сказано в Глава 7.

Установим пакет:

```
make install
```

Сделаем установленную библиотеку доступной для записи, чтобы позже мы могли удалить отладочную информацию:

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

Установим заголовочные файлы Tcl. Следующий пакет, Expect, требует их для сборки.

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

Создадим необходимую символическую ссылку:

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

5.11.2. Содержание Tcl

Установленные программы:	tclsh (ссылка на tclsh8.5) и tclsh8.5
Установленная библиотека:	libtcl8.5.so, libtclstub8.5.a

Краткое описание

tclsh8.5	Командная оболочка Tcl
tclsh	Ссылка на tclsh8.5
`libtcl8.5.so`	Библиотека Tcl
`libtclstub8.5.a`	Базовая библиотека Tcl

5.12. Expect-5.45

Пакет Expect содержит программу для добавления диалогов в интерактивные программы.

Приблизительное время сборки:	0.1 SBU
Требует свободного места на диске:	4.1 MB

5.12.1. Установка Expect

Сначала заставим скрипт configure использовать `/bin/stty` вместо `/usr/local/bin/stty`, который он может найти на хост-системе. Это гарантирует, что наши утилиты тестирования будут работать до окончания построения финальной системы:

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

Теперь подготовим Expect к компиляции:

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

Значение опций configure:

`--with-tcl=/tools/lib`
Это указывает скрипту configure искать Tcl в директории наших временных инструментов; в противном случае он может подхватить установку Tcl хост-системы.

`--with-tclinclude=/tools/include`
Говорит Expect, где следует искать заголовочные файлы Tcl.

Скомпилируем пакет:

```
make
```

Компиляция завершена. Как говорилось ранее, необязательно выполнять тесты для временных инструментов в этой главе. Чтобы все же запустить тестирование Expect, выполните следующую команду:

```
make test
```

Тестирование Expect может закончиться неудачно из-за некоторых условий хост-системы, которые мы не можем проконтролировать. Поэтому ошибки при проверке не являются сюрпризом, и не следует считать их критичными.

Установим пакет:

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

Значение параметров make:

`SCRIPTS=""`
Это предотвращает установку некоторых скриптов Expect, которые не нужны нам.

5.12.2. Содержимое Expect

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

Краткое описание

expect	Взаимодействует с другими интерактивными программами в соответствии со скриптом
`libexpect-5.45.a`	Содержит функции, которые позволяют использовать Expect как расширение Tcl или напрямую из C или C++ (без Tcl)

5.13. DejaGNU-1.4.4

Пакет DejaGNU содержит утилиты для тестирования других программ.

Приблизительное время сборки: less than 0.1 SBU
Требует свободного места на диске: 6.1 MB

5.13.1. Установка DejaGNU

Последняя официальная версия этого пакета была выпущена в 2004 году. Стоит применить некоторые исправления, накопившиеся с тех пор:

```
patch -Np1 -i ../dejagnu-1.4.4-consolidated-1.patch
```

Подготовим DejaGNU для компиляции:

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

Соберем и установим пакет:

```
make install
```

Чтобы проверить результат, выполните:

```
make check
```

5.13.2. Содержание DejaGNU

Установленная программа: runtest

Краткое описание

runtest Скрипт-обертка, который ищет подходящую оболочку **expect** и затем запускает DejaGNU

5.14. Ncurses-5.7

The Ncurses package contains libraries for terminal-independent handling of character screens.

Приблизительное время сборки:	0.7 SBU
Требует свободного места на диске:	30 MB

5.14.1. Установка Ncurses

Подготовим Ncurses к компиляции:

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

Значение ключей configure:

`--without-ada`
Запрещает Ncurses собирать поддержку для компилятора Ada, который может присутствовать на хост-системе, но не будет доступен после того, как мы войдем в окружение **chroot**.

`--enable-overwrite`
Это указывает Ncurses установить заголовочные файлы в `/tools/include` вместо `/tools/include/ncurses`, чтобы другие пакеты могли успешно найти их.

Скомпилируем пакет:

```
make
```

Этот пакет имеет набор тестов, но они могут быть запущены только после его установки. Тесты располагаются в директории `test/`. Более подробная информация находится в файле `README` в этой же директории.

Установим пакет:

```
make install
```

Подробная информация об этом пакете расположена в Раздел 6.19.2, «Contents of Ncurses.»

5.15. Bash-4.2

The Bash package contains the Bourne-Again SHell.

Приблизительное время сборки:	0.5 SBU
Требует свободного места на диске:	35 MB

5.15.1. Установка Bash

Подготовим Bash к компиляции:

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

Значение опций configure:

`--without-bash-malloc`
Эта опция отключает использование встроенной функции выделения памяти Bash (`malloc`), которая часто вызывает ошибки сегментирования. При отключении этой опции Bash будет использовать функцию `malloc` из Glibc, которая гораздо более надежна.

Скомпилируем пакет:

```
make
```

Компиляция завершена. Как говорилось ранее, необязательно выполнять тесты для временных инструментов в этой главе. Чтобы все же запустить тестирование Bash, выполните следующую команду:

```
make tests
```

Установим пакет:

```
make install
```

Сделаем ссылку для программ, которые используют **sh** как оболочку:

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

Подробная информация об этом пакете расположена в Раздел 6.29.2, «Contents of Bash.»

5.16. Bzip2-1.0.6

The Bzip2 package contains programs for compressing and decompressing files. Compressing text files with **bzip2** yields a much better compression percentage than with the traditional **gzip**.

Приблизительное время сборки:	less than 0.1 SBU
Требует свободного места на диске:	4.8 MB

5.16.1. Установка Bzip2

Пакет Bzip2 не содержит скрипта **configure**. Скомпилируем его:

```
make
```

Установим пакет:

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

Подробная информация об этом пакете расположена в Раздел 6.36.2, «Contents of Bzip2.»

5.17. Coreutils-8.10

The Coreutils package contains utilities for showing and setting the basic system characteristics.

Приблизительное время сборки:	0.7 SBU
Требует свободного места на диске:	88 MB

5.17.1. Установка Coreutils

Подготовим Coreutils к компиляции:

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

Значение опций configure:

`--enable-install-program=hostname`
Это указывает собрать и установить утилиту **hostname** – она отключена по умолчанию, но требуется Perl для выполнения тестов.

Скомпилируем пакет:

```
make
```

Компиляция завершена. Как говорилось ранее, необязательно выполнять тесты для временных инструментов в этой главе. Чтобы все же запустить тестирование Coreutils, выполните следующую команду:

```
make RUN_EXPENSIVE_TESTS=yes check
```

Параметр *RUN_EXPENSIVE_TESTS=yes* указывает выполнить несколько дополнительных тестов, которые являются достаточно дорогостоящими (в плане мощности процессора CPU и использования памяти) на некоторых платформах, но обычно не являются проблемой для Linux.

Установим пакет:

```
make install
```

Вышеприведенная команда не установит `su`, поскольку программа не может быть установлена с флагом "setuid root" от имени непривелегированного пользователя. Установив ее вручную с другим именем, мы сможем выполнять тесты от непривелегированного пользователя в конечной системе и сохранить возможно полезную **su** из хост-системы первой в PATH. Установим ее:

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

Подробная информация об этом пакете расположена в Раздел 6.22.2, «Contents of Coreutils.»

5.18. Diffutils-3.0

The Diffutils package contains programs that show the differences between files or directories.

Приблизительное время сборки:	0.1 SBU
Требует свободного места на диске:	6.1 MB

5.18.1. Установка Diffutils

Подготовим Diffutils к компиляции:

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

Скомпилируем пакет:

```
make
```

Компиляция завершена. Как говорилось ранее, необязательно выполнять тесты для временных инструментов в этой главе. Чтобы все же запустить тестирование Diffutils, выполните следующую команду:

```
make check
```

Установим пакет:

```
make install
```

Подробная информация об этом пакете расположена в Раздел 6.37.2, «Contents of Diffutils.»

5.19. File-5.05

The File package contains a utility for determining the type of a given file or files.

Приблизительное время сборки:	0.2 SBU
Требует свободного места на диске:	9.5 MB

5.19.1. Installation of File

Prepare File for compilation:

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

Compile the package:

```
make
```

Compilation is now complete. As discussed earlier, running the test suite is not mandatory for the temporary tools here in this chapter. To run the File test suite anyway, issue the following command:

```
make check
```

Install the package:

```
make install
```

Details on this package are located in Раздел 6.39.2, «Contents of File.»

5.20. Findutils-4.4.2

The Findutils package contains programs to find files. These programs are provided to recursively search through a directory tree and to create, maintain, and search a database (often faster than the recursive find, but unreliable if the database has not been recently updated).

Приблизительное время сборки:	0.3 SBU
Требует свободного места на диске:	20 MB

5.20.1. Установка Findutils

Подготовим Findutils к компиляции:

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

Компилируем пакет:

```
make
```

Компиляция завершена. Как говорилось ранее, необязательно выполнять тесты для временных инструментов в этой главе. Чтобы все же запустить тестирование Findutils, выполните следующую команду:

```
make check
```

Установим пакет:

```
make install
```

Подробная информация об этом пакете расположена в Раздел 6.40.2, «Contents of Findutils.»

5.21. Gawk-3.1.8

The Gawk package contains programs for manipulating text files.

Приблизительное время сборки:	0.2 SBU
Требует свободного места на диске:	19 MB

5.21.1. Установка Gawk

Подготовим Gawk к компиляции:

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

Скомпилируем пакет:

```
make
```

Компиляция завершена. Как говорилось ранее, необязательно выполнять тесты для временных инструментов в этой главе. Чтобы все же запустить тестирование Gawk, выполните следующую команду:

```
make check
```

Установим пакет:

```
make install
```

Подробная информация об этом пакете расположена в Раздел 6.38.2, «Contents of Gawk.»

5.22. Gettext-0.18.1.1

The Gettext package contains utilities for internationalization and localization. These allow programs to be compiled with NLS (Native Language Support), enabling them to output messages in the user's native language.

Приблизительное время сборки:	0.8 SBU
Требует свободного места на диске:	82 MB

5.22.1. Установка Gettext

Для нашего временного инструментария необходима только одна утилита из пакета Gettext.

Подготовим Gettext к компиляции:

```
cd gettext-tools
./configure --prefix=/tools --disable-shared
```

Значение ключа configure:

`--disable-shared`
Нам в данный момент не требуется устанавливать разделяемые библиотеки Gettext, поэтому нет нужды и собирать их.

Скомпилируем пакет:

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

Поскольку была скомпилирована только одна утилита, невозможно запустить тестирование пакета без компиляции дополнительных библиотек из пакета Gettext. Не рекомендуется пытаться запустить тесты на этой стадии.

Установим утилиту **msgfmt**:

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

Подробная информация об этом пакете расположена в Раздел 6.42.2, «Contents of Gettext.»

5.23. Grep-2.7

The Grep package contains programs for searching through files.

Приблизительное время сборки:	0.1 SBU
Требует свободного места на диске:	6.7 MB

5.23.1. Установка Grep

Подготовим Grep к компиляции:

```
./configure --prefix=/tools \
    --disable-perl-regexp
```

Значение ключей configure:

`--disable-perl-regexp`
Это запрещает программе **grep** использовать библиотеку Perl Compatible Regular Expression (PCRE), которая может присутствовать на хост-системе, но не будет доступна после входа в **chroot**-окружение.

Скомпилируем пакет:

```
make
```

Компиляция завершена. Как говорилось ранее, необязательно выполнять тесты для временных инструментов в этой главе. Чтобы все же запустить тестирование Grep, выполните следующую команду:

```
make check
```

Установим пакет:

```
make install
```

Подробная информация об этом пакете расположена в Раздел 6.27.2, «Contents of Grep.»

5.24. Gzip-1.4

The Gzip package contains programs for compressing and decompressing files.

Приблизительное время сборки:	less than 0.1 SBU
Требует свободного места на диске:	3.3 MB

5.24.1. Установка Gzip

Подготовим Gzip к компиляции:

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

Скомпилируем пакет:

```
make
```

Компиляция завершена. Как говорилось ранее, необязательно выполнять тесты для временных инструментов в этой главе. Чтобы все же запустить тестирование Gzip, выполните следующую команду:

```
make check
```

Установим пакет:

```
make install
```

Подробная информация об этом пакете расположена в Раздел 6.45.2, «Contents of Gzip.»

5.25. M4-1.4.15

The M4 package contains a macro processor.

Приблизительное время сборки:	0.2 SBU
Требует свободного места на диске:	11.6 MB

5.25.1. Установка M4

Подготовим M4 к компиляции:

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

Скомпилируем пакет:

```
make
```

Компиляция завершена. Как говорилось ранее, необязательно выполнять тесты для временных инструментов в этой главе. Чтобы все же запустить тестирование M4, выполните следующую команду:

```
make check
```

Установим пакет:

```
make install
```

Подробная информация об этом пакете расположена в Раздел 6.24.2, «Contents of M4.»

5.26. Make-3.82

The Make package contains a program for compiling packages.

Приблизительное время сборки:	0.1 SBU
Требует свободного места на диске:	9.6 MB

5.26.1. Установка Make

Подготовим Make к компиляции:

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

Скомпилируем пакет:

```
make
```

Компиляция завершена. Как говорилось ранее, необязательно выполнять тесты для временных инструментов в этой главе. Чтобы все же запустить тестирование Make, выполните следующую команду:

```
make check
```

Установим пакет:

```
make install
```

Подробная информация об этом пакете расположена в Раздел 6.49.2, «Contents of Make.»

5.27. Patch-2.6.1

The Patch package contains a program for modifying or creating files by applying a «patch» file typically created by the **diff** program.

Приблизительное время сборки:	less than 0.1 SBU
Требует свободного места на диске:	1.9 MB

5.27.1. Установка Patch

Подготовим Patch к компиляции:

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

Скомпилируем пакет:

```
make
```

Компиляция завершена. Как говорилось ранее, необязательно выполнять тесты для временных инструментов в этой главе. Чтобы все же запустить тестирование Patch, выполните следующую команду:

```
make check
```

Установим пакет:

```
make install
```

Подробная информация об этом пакете расположена в Раздел 6.53.2, «Contents of Patch.»

5.28. Perl-5.12.3

The Perl package contains the Practical Extraction and Report Language.

Приблизительное время сборки:	0.8 SBU
Требует свободного места на диске:	106 MB

5.28.1. Установка Perl

First apply the following patch to adapt some hard-wired paths to the C library:

```
patch -Np1 -i ../perl-5.12.3-libc-1.patch
```

Подготовим Perl к компиляции (make sure to get the 'Data/Dumper Fcntl IO' part of the command correct—they are all letters):

```
sh Configure -des -Dprefix=/tools \
                  -Dstatic_ext='Data/Dumper Fcntl IO'
```

The meaning of the configure options:

`-Dstatic_ext='Data/Dumper Fcntl IO'`
: This tells Perl to build the minimum set of static extensions needed for installing and testing the Coreutils and Glibc packages in the next chapter.

Only a few of the utilities contained in this package, and one of its libraries, need to be built:

```
make perl utilities ext/Errno/pm_to_blib
```

Although Perl comes with a test suite, it is not recommended to run it at this point. Only part of Perl was built and running **make test** now will cause the rest of Perl to be built as well, which is unnecessary at this point. The test suite can be run in the next chapter if desired.

Install these tools and their libraries:

```
cp -v perl pod/pod2man /tools/bin
mkdir -pv /tools/lib/perl5/5.12.3
cp -Rv lib/* /tools/lib/perl5/5.12.3
```

Подробная информация об этом пакете расположена в Раздел 6.33.2, «Contents of Perl.»

5.29. Sed-4.2.1

The Sed package contains a stream editor.

Приблизительное время сборки:	0.1 SBU
Требует свободного места на диске:	8.0 MB

5.29.1. Установка Sed

Подготовим Sed к компиляции:

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

Скомпилируем пакет:

```
make
```

Компиляция завершена. Как говорилось ранее, необязательно выполнять тесты для временных инструментов в этой главе. Чтобы все же запустить тестирование Sed, выполните следующую команду:

```
make check
```

Установим пакет:

```
make install
```

Подробная информация об этом пакете расположена в Раздел 6.17.2, «Contents of Sed.»

5.30. Tar-1.25

The Tar package contains an archiving program.

Приблизительное время сборки:	0.3 SBU
Требует свободного места на диске:	20.9 MB

5.30.1. Установка Tar

Подготовим Tar к компиляции:

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

Скомпилируем пакет:

```
make
```

Компиляция завершена. Как говорилось ранее, необязательно выполнять тесты для временных инструментов в этой главе. Чтобы все же запустить тестирование Tar, выполните следующую команду:

```
make check
```

Установим пакет:

```
make install
```

Подробная информация об этом пакете расположена в Раздел 6.58.2, «Contents of Tar.»

5.31. Texinfo-4.13a

The Texinfo package contains programs for reading, writing, and converting info pages.

Приблизительное время сборки:	0.2 SBU
Требует свободного места на диске:	20 MB

5.31.1. Установка Texinfo

Подготовим Texinfo к компиляции:

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

Скомпилируем пакет:

```
make
```

Компиляция завершена. Как говорилось ранее, необязательно выполнять тесты для временных инструментов в этой главе. Чтобы все же запустить тестирование Texinfo, выполните следующую команду:

```
make check
```

Установим пакет:

```
make install
```

Подробная информация об этом пакете расположена в Раздел 6.59.2, «Contents of Texinfo.»

5.32. Xz-5.0.1

The Xz package contains programs for compressing and decompressing files. It provides capabilities for the lzma and the newer xz compression formats. Compressing text files with **xz** yields a better compression percentage than with the traditional **gzip** or **bzip2** commands.

Приблизительное время сборки:	0.3 SBU
Требует свободного места на диске:	14 MB

5.32.1. Installation of Xz-Utils

Prepare Xz for compilation:

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

Compile the package:

```
make
```

Compilation is now complete. As discussed earlier, running the test suite is not mandatory for the temporary tools here in this chapter. To run the Xz test suite anyway, issue the following command:

```
make check
```

Install the package:

```
make install
```

Details on this package are located in Раздел 6.50.2, «Contents of Xz.»

5.33. Очистка

Шаги, предлагаемые в этой секции, не являются обязательными, однако, если раздел LFS очень мал, неплохо будет узнать, какие ненужные элементы можно удалить. Собранные исполняемые файлы и библиотеки содержат около 70 MB ненужной на данном этапе отладочной информации. Удалите ее командами:

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

Эти команды пропустят некоторые файлы, сообщая что не могут распознать их формат. Большинство из них являются скриптами, а не двоичными файлами.

Будьте внимательны и *не* используйте параметр *--strip-unneeded* при обработке библиотек. Статические библиотеки будут полностью испорчены и весь инструментарий прийдется собирать заново.

Чтобы освободить еще около 25 MB, удалите документацию:

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

На данном этапе у Вас должно остаться как минимум 850 MB свободного места в разделе $LFS. Именно столько потребуется для сборки и установки Glibc в следующей главе. Если Вы сможете собрать Glibc, Вам хватит места и на все остальное.

5.34. Смена владельца

Замечание

Все следующие команды в этой книге должны быть выполнены от имени пользователя root, а не lfs. Поэтому дважды проверьте, что переменная $LFS объявлена в окружении root.

В данный момент директория $LFS/tools принадлежит пользователю lfs , который существует только на хост-системе. Если директория $LFS/tools будет сохранена для последующего использования, файлы будут принадлежать идентификатору пользователя, которому не соответствует ни одна учетная запись. Это очень опасно, поскольку позже созданная учетная запись пользователя может получить такой же идентификатор, и сможет сделать с файлами и папками в директории $LFS/tools что угодно, возможно, даже полностью разрушить их.

Чтобы предотвратить это, Вы можете добавить пользователя lfs в новую систему LFS позже, при создании файла /etc/passwd, позаботившись о присвоении ему тех же идентификаторов пользователя и группы, что и на хост-системе. Но еще лучше просто сменить владельца директории $LFS/tools на пользователя root выполнив следующую команду:

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

Хотя директория $LFS/tools может быть удалена после завершения сборки LFS-системы, ее можно оставить для сборки последующих LFS-систем *той же версии книги*. Как лучше всего сохранить $LFS/tools - Ваше личное дело.

Предостережение

Если Вы собираетесь сохранить временные инструменты для использования при сборке будущих LFS-систем, необходимо *сейчас* сохранить их. Последующие команды в главе 6 изменят их, сделав непригодными для будущих сборок.

Часть III. Сборка системы LFS

Глава 6. Installing Basic System Software

6.1. Introduction

In this chapter, we enter the building site and start constructing the LFS system in earnest. That is, we chroot into the temporary mini Linux system, make a few final preparations, and then begin installing the packages.

The installation of this software is straightforward. Although in many cases the installation instructions could be made shorter and more generic, we have opted to provide the full instructions for every package to minimize the possibilities for mistakes. The key to learning what makes a Linux system work is to know what each package is used for and why you (or the system) may need it.

We do not recommend using optimizations. They can make a program run slightly faster, but they may also cause compilation difficulties and problems when running the program. If a package refuses to compile when using optimization, try to compile it without optimization and see if that fixes the problem. Even if the package does compile when using optimization, there is the risk it may have been compiled incorrectly because of the complex interactions between the code and build tools. Also note that the `-march` and `-mtune` options using values not specified in the book have not been tested. This may cause problems with the toolchain packages (Binutils, GCC and Glibc). The small potential gains achieved in using compiler optimizations are often outweighed by the risks. First-time builders of LFS are encouraged to build without custom optimizations. The subsequent system will still run very fast and be stable at the same time.

The order that packages are installed in this chapter needs to be strictly followed to ensure that no program accidentally acquires a path referring to `/tools` hard-wired into it. For the same reason, do not compile separate packages in parallel. Compiling in parallel may save time (especially on dual-CPU machines), but it could result in a program containing a hard-wired path to `/tools`, which will cause the program to stop working when that directory is removed.

Before the installation instructions, each installation page provides information about the package, including a concise description of what it contains, approximately how long it will take to build, and how much disk space is required during this building process. Following the installation instructions, there is a list of programs and libraries (along with brief descriptions of these) that the package installs.

Замечание

The SBU values and required disk space includes test suite data for all applicable packages in Chapter 6.

6.2. Preparing Virtual Kernel File Systems

Various file systems exported by the kernel are used to communicate to and from the kernel itself. These file systems are virtual in that no disk space is used for them. The content of the file systems resides in memory.

Begin by creating directories onto which the file systems will be mounted:

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

6.2.1. Creating Initial Device Nodes

When the kernel boots the system, it requires the presence of a few device nodes, in particular the `console` and `null` devices. The device nodes must be created on the hard disk so that they are available before **udevd** has been started, and additionally when Linux is started with *`init=/bin/bash`*. Create the devices by running the following commands:

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

6.2.2. Mounting and Populating /dev

The recommended method of populating the `/dev` directory with devices is to mount a virtual filesystem (such as `tmpfs`) on the `/dev` directory, and allow the devices to be created dynamically on that virtual filesystem as they are detected or accessed. Device creation is generally done during the boot process by Udev. Since this new system does not yet have Udev and has not yet been booted, it is necessary to mount and populate `/dev` manually. This is accomplished by bind mounting the host system's `/dev` directory. A bind mount is a special type of mount that allows you to create a mirror of a directory or mount point to some other location. Use the following command to achieve this:

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

6.2.3. Mounting Virtual Kernel File Systems

Now mount the remaining virtual kernel filesystems:

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

6.3. Package Management

Package Management is an often requested addition to the LFS Book. A Package Manager allows tracking the installation of files making it easy to remove and upgrade packages. As well as the binary and library files, a package manager will handle the installation of configuration files. Before you begin to wonder, NO—this section will not talk about nor recommend any particular package manager. What it provides is a roundup of the more popular techniques and how they work. The perfect package manager for you may be among these techniques or may be a combination of two or more of these techniques. This section briefly mentions issues that may arise when upgrading packages.

Some reasons why no package manager is mentioned in LFS or BLFS include:

- Dealing with package management takes the focus away from the goals of these books—teaching how a Linux system is built.
- There are multiple solutions for package management, each having its strengths and drawbacks. Including one that satisfies all audiences is difficult.

There are some hints written on the topic of package management. Visit the *Hints Project* and see if one of them fits your need.

6.3.1. Upgrade Issues

A Package Manager makes it easy to upgrade to newer versions when they are released. Generally the instructions in the LFS and BLFS Book can be used to upgrade to the newer versions. Here are some points that you should be aware of when upgrading packages, especially on a running system.

- If one of the toolchain packages (Glibc, GCC or Binutils) needs to be upgraded to a newer minor version, it is safer to rebuild LFS. Though you *may* be able to get by rebuilding all the packages in their dependency order, we do not recommend it. For example, if glibc-2.2.x needs to be updated to glibc-2.3.x, it is safer to rebuild. For micro version updates, a simple reinstallation usually works, but is not guaranteed. For example, upgrading from glibc-2.3.4 to glibc-2.3.5 will not usually cause any problems.
- If a package containing a shared library is updated, and if the name of the library changes, then all the packages dynamically linked to the library need to be recompiled to link against the newer library. (Note that there is no correlation between the package version and the name of the library.) For example, consider a package foo-1.2.3 that installs a shared library with name `libfoo.so.1`. Say you upgrade the package to a newer version foo-1.2.4 that installs a shared library with name `libfoo.so.2`. In this case, all packages that are dynamically linked to `libfoo.so.1` need to be recompiled to link against `libfoo.so.2`. Note that you should not remove the previous libraries until the dependent packages are recompiled.

6.3.2. Package Management Techniques

The following are some common package management techniques. Before making a decision on a package manager, do some research on the various techniques, particularly the drawbacks of the particular scheme.

6.3.2.1. It is All in My Head!

Yes, this is a package management technique. Some folks do not find the need for a package manager because they know the packages intimately and know what files are installed by each package. Some users also do not need any package management because they plan on rebuilding the entire system when a package is changed.

6.3.2.2. Install in Separate Directories

This is a simplistic package management that does not need any extra package to manage the installations. Each package is installed in a separate directory. For example, package foo-1.1 is installed in `/usr/pkg/foo-1.1` and a symlink is made from `/usr/pkg/foo` to `/usr/pkg/foo-1.1`. When installing a new version foo-1.2, it is installed in `/usr/pkg/foo-1.2` and the previous symlink is replaced by a symlink to the new version.

Environment variables such as `PATH`, `LD_LIBRARY_PATH`, `MANPATH`, `INFOPATH` and `CPPFLAGS` need to be expanded to include `/usr/pkg/foo`. For more than a few packages, this scheme becomes unmanageable.

6.3.2.3. Symlink Style Package Management

This is a variation of the previous package management technique. Each package is installed similar to the previous scheme. But instead of making the symlink, each file is symlinked into the `/usr` hierarchy. This removes the need to expand the environment

variables. Though the symlinks can be created by the user to automate the creation, many package managers have been written using this approach. A few of the popular ones include Stow, Epkg, Graft, and Depot.

The installation needs to be faked, so that the package thinks that it is installed in `/usr` though in reality it is installed in the `/usr/pkg` hierarchy. Installing in this manner is not usually a trivial task. For example, consider that you are installing a package libfoo-1.1. The following instructions may not install the package properly:

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

The installation will work, but the dependent packages may not link to libfoo as you would expect. If you compile a package that links against libfoo, you may notice that it is linked to `/usr/pkg/libfoo/1.1/lib/libfoo.so.1` instead of `/usr/lib/libfoo.so.1` as you would expect. The correct approach is to use the `DESTDIR` strategy to fake installation of the package. This approach works as follows:

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

Most packages support this approach, but there are some which do not. For the non-compliant packages, you may either need to manually install the package, or you may find that it is easier to install some problematic packages into `/opt`.

6.3.2.4. Timestamp Based

In this technique, a file is timestamped before the installation of the package. After the installation, a simple use of the **find** command with the appropriate options can generate a log of all the files installed after the timestamp file was created. A package manager written with this approach is install-log.

Though this scheme has the advantage of being simple, it has two drawbacks. If, during installation, the files are installed with any timestamp other than the current time, those files will not be tracked by the package manager. Also, this scheme can only be used when one package is installed at a time. The logs are not reliable if two packages are being installed on two different consoles.

6.3.2.5. Tracing Installation Scripts

In this approach, the commands that the installation scripts perform are recorded. There are two techniques that one can use:

The `LD_PRELOAD` environment variable can be set to point to a library to be preloaded before installation. During installation, this library tracks the packages that are being installed by attaching itself to various executables such as **cp**, **install**, **mv** and tracking the system calls that modify the filesystem. For this approach to work, all the executables need to be dynamically linked without the suid or sgid bit. Preloading the library may cause some unwanted side-effects during installation. Therefore, it is advised that one performs some tests to ensure that the package manager does not break anything and logs all the appropriate files.

The second technique is to use **strace**, which logs all system calls made during the execution of the installation scripts.

6.3.2.6. Creating Package Archives

In this scheme, the package installation is faked into a separate tree as described in the Symlink style package management. After the installation, a package archive is created using the installed files. This archive is then used to install the package either on the local machine or can even be used to install the package on other machines.

This approach is used by most of the package managers found in the commercial distributions. Examples of package managers that follow this approach are RPM (which, incidentally, is required by the *Linux Standard Base Specification*), pkg-utils, Debian's apt, and Gentoo's Portage system. A hint describing how to adopt this style of package management for LFS systems is located at *http://www.linuxfromscratch.org/hints/downloads/files/fakeroot.txt*.

Creation of package files that include dependency information is complex and is beyond the scope of LFS.

Slackware uses a **tar** based system for package archives. This system purposely does not handle package dependencies as more complex package managers do. For details of Slackware package management, see *http://www.slackbook.org/html/package-management.html*.

6.3.2.7. User Based Management

This scheme, unique to LFS, was devised by Matthias Benkmann, and is available from the *Hints Project*. In this scheme, each package is installed as a separate user into the standard locations. Files belonging to a package are easily identified by checking the user ID. The features and shortcomings of this approach are too complex to describe in this section. For the details please see the hint at *http://www.linuxfromscratch.org/hints/downloads/files/more_control_and_pkg_man.txt*.

6.3.3. Deploying LFS on Multiple Systems

One of the advantages of an LFS system is that there are no files that depend on the position of files on a disk system. Cloning an LFS build to another computer with an architecture similar to the base system is as simple as using **tar** on the LFS partition that contains the root directory (about 250MB uncompressed for a base LFS build), copying that file via network transfer or CD-ROM to the new system and expanding it. From that point, a few configuration files will have to be changed. Configuration files that may need to be updated include: `/etc/hosts`, `/etc/fstab`, `/etc/passwd`, `/etc/group`, `/etc/shadow`, `/etc/ld.so.conf`, `/etc/scsi_id.config`, `/etc/sysconfig/network` and `/etc/sysconfig/network-devices/ifconfig.eth0/ipv4`.

A custom kernel may need to be built for the new system depending on differences in system hardware and the original kernel configuration.

Finally the new system has to be made bootable via Раздел 8.4, «Using GRUB to Set Up the Boot Process».

6.4. Entering the Chroot Environment

It is time to enter the chroot environment to begin building and installing the final LFS system. As user `root`, run the following command to enter the realm that is, at the moment, populated with only the temporary tools:

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

The *-i* option given to the **env** command will clear all variables of the chroot environment. After that, only the `HOME`, `TERM`, `PS1`, and `PATH` variables are set again. The *TERM=$TERM* construct will set the `TERM` variable inside chroot to the same value as outside chroot. This variable is needed for programs like **vim** and **less** to operate properly. If other variables are needed, such as `CFLAGS` or `CXXFLAGS`, this is a good place to set them again.

From this point on, there is no need to use the `LFS` variable anymore, because all work will be restricted to the LFS file system. This is because the Bash shell is told that `$LFS` is now the root (`/`) directory.

Notice that `/tools/bin` comes last in the `PATH`. This means that a temporary tool will no longer be used once its final version is installed. This occurs when the shell does not «remember» the locations of executed binaries—for this reason, hashing is switched off by passing the *+h* option to **bash**.

Note that the **bash** prompt will say `I have no name!` This is normal because the `/etc/passwd` file has not been created yet.

Замечание

It is important that all the commands throughout the remainder of this chapter and the following chapters are run from within the chroot environment. If you leave this environment for any reason (rebooting for example), ensure that the virtual kernel filesystems are mounted as explained in Раздел 6.2.2, «Mounting and Populating /dev» and Раздел 6.2.3, «Mounting Virtual Kernel File Systems» and enter chroot again before continuing with the installation.

6.5. Creating Directories

It is time to create some structure in the LFS file system. Create a standard directory tree by issuing the following commands:

```
mkdir -pv /{bin,boot,etc/opt,home,lib,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/{doc,info,locale,man}
mkdir -v  /usr/{,local/}share/{misc,terminfo,zoneinfo}
mkdir -pv /usr/{,local/}share/man/man{1..8}
for dir in /usr /usr/local; do
  ln -sv share/{man,doc,info} $dir
done
case $(uname -m) in
 x86_64) ln -sv lib /lib64 && ln -sv lib /usr/lib64 ;;
esac
mkdir -v /var/{lock,log,mail,run,spool}
mkdir -pv /var/{opt,cache,lib/{misc,locate},local}
```

Directories are, by default, created with permission mode 755, but this is not desirable for all directories. In the commands above, two changes are made—one to the home directory of user `root`, and another to the directories for temporary files.

The first mode change ensures that not just anybody can enter the `/root` directory—the same as a normal user would do with his or her home directory. The second mode change makes sure that any user can write to the `/tmp` and `/var/tmp` directories, but cannot remove another user's files from them. The latter is prohibited by the so-called «sticky bit,» the highest bit (1) in the 1777 bit mask.

6.5.1. FHS Compliance Note

The directory tree is based on the Filesystem Hierarchy Standard (FHS) (available at *http://www.pathname.com/fhs/*). In addition to the FHS, we create compatibility symlinks for the `man`, `doc`, and `info` directories since many packages still try to install their documentation into `/usr/<directory>` or `/usr/local/<directory>` as opposed to `/usr/share/<directory>` or `/usr/local/share/<directory>`. The FHS also stipulates the existence of `/usr/local/games` and `/usr/share/games`. The FHS is not precise as to the structure of the `/usr/local/share` subdirectory, so we create only the directories that are needed. However, feel free to create these directories if you prefer to conform more strictly to the FHS.

6.6. Creating Essential Files and Symlinks

Some programs use hard-wired paths to programs which do not exist yet. In order to satisfy these programs, create a number of symbolic links which will be replaced by real files throughout the course of this chapter after the software has been installed:

```
ln -sv /tools/bin/{bash,cat,echo,pwd,stty} /bin
ln -sv /tools/bin/perl /usr/bin
ln -sv /tools/lib/libgcc_s.so{,.1} /usr/lib
ln -sv /tools/lib/libstdc++.so{,.6} /usr/lib
ln -sv bash /bin/sh
```

A proper Linux system maintains a list of the mounted file systems in the file `/etc/mtab`. Normally, this file would be created when we mount a new file system. Since we will not be mounting any file systems inside our chroot environment, create an empty file for utilities that expect the presence of `/etc/mtab`:

```
touch /etc/mtab
```

In order for user `root` to be able to login and for the name «root» to be recognized, there must be relevant entries in the `/etc/passwd` and `/etc/group` files.

Create the `/etc/passwd` file by running the following command:

```
cat > /etc/passwd << "EOF"
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/dev/null:/bin/false
nobody:x:99:99:Unprivileged User:/dev/null:/bin/false
EOF
```

The actual password for `root` (the «x» used here is just a placeholder) will be set later.

Create the `/etc/group` file by running the following command:

```
cat > /etc/group << "EOF"
root:x:0:
bin:x:1:
sys:x:2:
kmem:x:3:
tty:x:4:
tape:x:5:
daemon:x:6:
floppy:x:7:
disk:x:8:
lp:x:9:
dialout:x:10:
audio:x:11:
video:x:12:
utmp:x:13:
usb:x:14:
cdrom:x:15:
mail:x:34:
nogroup:x:99:
EOF
```

The created groups are not part of any standard—they are groups decided on in part by the requirements of the Udev configuration in this chapter, and in part by common convention employed by a number of existing Linux distributions. The Linux Standard Base (LSB, available at *http://www.linuxbase.org*) recommends only that, besides the group `root` with a Group ID (GID) of 0, a group `bin` with a GID of 1 be present. All other group names and GIDs can be chosen freely by the system administrator since well-written programs do not depend on GID numbers, but rather use the group's name.

To remove the «I have no name!» prompt, start a new shell. Since a full Glibc was installed in Глава 5 and the `/etc/passwd` and `/etc/group` files have been created, user name and group name resolution will now work:

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

Note the use of the *+h* directive. This tells **bash** not to use its internal path hashing. Without this directive, **bash** would remember the paths to binaries it has executed. To ensure the use of the newly compiled binaries as soon as they are installed, the *+h* directive will be used for the duration of this chapter.

The **login**, **agetty**, and **init** programs (and others) use a number of log files to record information such as who was logged into the system and when. However, these programs will not write to the log files if they do not already exist. Initialize the log files and give them proper permissions:

```
touch /var/run/utmp /var/log/{btmp,lastlog,wtmp}
chgrp -v utmp /var/run/utmp /var/log/lastlog
chmod -v 664 /var/run/utmp /var/log/lastlog
```

The `/var/run/utmp` file records the users that are currently logged in. The `/var/log/wtmp` file records all logins and logouts. The `/var/log/lastlog` file records when each user last logged in. The `/var/log/btmp` file records the bad login attempts.

6.7. Linux-2.6.37 API Headers

The Linux API Headers expose the kernel's API for use by Glibc.

Приблизительное время сборки:	0.1 SBU
Требует свободного места на диске:	485 MB

6.7.1. Installation of Linux API Headers

The Linux kernel needs to expose an Application Programming Interface (API) for the system's C library (Glibc in LFS) to use. This is done by way of sanitizing various C header files that are shipped in the Linux kernel source tarball.

Make sure there are no stale files and dependencies lying around from previous activity:

```
make mrproper
```

Now test and extract the user-visible kernel headers from the source. They are placed in an intermediate local directory and copied to the needed location because the extraction process removes any existing files in the target directory. There are also some hidden files used by the kernel developers and not needed by LFS that are removed from the intermediate directory.

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

6.7.2. Contents of Linux API Headers

Installed headers:	/usr/include/asm/*.h, /usr/include/asm-generic/*.h, /usr/include/drm/*.h, /usr/include/linux/*.h, /usr/include/mtd/*.h, /usr/include/rdma/*.h, /usr/include/scsi/*.h, /usr/include/sound/*.h, /usr/include/video/*.h, /usr/include/xen/*.h
Installed directories:	/usr/include/asm, /usr/include/asm-generic, /usr/include/drm, /usr/include/linux, /usr/include/mtd, /usr/include/rdma, /usr/include/scsi, /usr/include/sound, /usr/include/video, /usr/include/xen

Short Descriptions

`/usr/include/asm/*.h`	The Linux API ASM Headers
`/usr/include/asm-generic/*.h`	The Linux API ASM Generic Headers
`/usr/include/drm/*.h`	The Linux API DRM Headers
`/usr/include/linux/*.h`	The Linux API Linux Headers
`/usr/include/mtd/*.h`	The Linux API MTD Headers
`/usr/include/rdma/*.h`	The Linux API RDMA Headers
`/usr/include/scsi/*.h`	The Linux API SCSI Headers

`/usr/include/sound/*.h`	The Linux API Sound Headers
`/usr/include/video/*.h`	The Linux API Video Headers
`/usr/include/xen/*.h`	The Linux API Xen Headers

6.8. Man-pages-3.32

The Man-pages package contains over 1,900 man pages.

Приблизительное время сборки: less than 0.1 SBU
Требует свободного места на диске: 21 MB

6.8.1. Installation of Man-pages

Install Man-pages by running:

```
make install
```

6.8.2. Contents of Man-pages

Installed files: various man pages

Short Descriptions

man pages	Describe C programming language functions, important device files, and significant configuration files

6.9. Glibc-2.13

The Glibc package contains the main C library. This library provides the basic routines for allocating memory, searching directories, opening and closing files, reading and writing files, string handling, pattern matching, arithmetic, and so on.

Приблизительное время сборки:	16.9 SBU
Требует свободного места на диске:	637 MB

6.9.1. Installation of Glibc

Замечание

Some packages outside of LFS suggest installing GNU libiconv in order to translate data from one encoding to another. The project's home page (*http://www.gnu.org/software/libiconv/*) says «This library provides an `iconv()` implementation, for use on systems which don't have one, or whose implementation cannot convert from/to Unicode.» Glibc provides an `iconv()` implementation and can convert from/to Unicode, therefore libiconv is not required on an LFS system.

The Glibc build system is self-contained and will install perfectly, even though the compiler specs file and linker are still pointing at `/tools`. The specs and linker cannot be adjusted before the Glibc install because the Glibc autoconf tests would give false results and defeat the goal of achieving a clean build.

When running **make install**, a script called `test-installation.pl` performs a small sanity test on our newly installed Glibc. However, because our toolchain still points to the `/tools` directory, the sanity test would be carried out against the wrong Glibc. We can force the script to check the Glibc we have just installed with the following:

```
DL=$(readelf -l /bin/sh | sed -n 's@.*interpret.*/tools\(.*\)]$@\1@p')
sed -i "s|libs -o|libs -L/usr/lib -Wl,-dynamic-linker=$DL -o|" \
        scripts/test-installation.pl
unset DL
```

In addition, there is a bug in the `test-installation.pl` script in that it tries to link a test program to a library that isn't installed by **make install**. Issue the following **sed** command to fix it:

```
sed -i -e 's/"db1"/& \&\& $name ne "nss_test1"/' scripts/test-installation.pl
```

The **ldd** shell script contains Bash-specific syntax. Change its default program interpreter to **/bin/bash** in case another **/bin/sh** is installed as described in the *Shells* chapter of the BLFS book:

```
sed -i 's|@BASH@|/bin/bash|' elf/ldd.bash.in
```

Fix a bug that prevents Glibc from building with GCC-4.5.2:

```
patch -Np1 -i ../glibc-2.13-gcc_fix-1.patch
```

Fix a stack imbalance that occurs under some conditions:

```
sed -i '195,213 s/PRIVATE_FUTEX/FUTEX_CLOCK_REALTIME/' \
nptl/sysdeps/unix/sysv/linux/x86_64/pthread_rwlock_timed{rd,wr}lock.S
```

The Glibc documentation recommends building Glibc outside of the source directory in a dedicated build directory:

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

As in Chapter 5, add the needed compiler flags to CFLAGS for x86 machines. Here, the optimization of the library is also set for the gcc compiler to enhance compilation speed (-pipe) and package performance (-O3).

```
case `uname -m` in
  i?86) echo "CFLAGS += -march=i486 -mtune=native -O3 -pipe" > configparms ;;
esac
```

Prepare Glibc for compilation:

```
../glibc-2.13/configure --prefix=/usr \
    --disable-profile --enable-add-ons \
    --enable-kernel=2.6.22.5 --libexecdir=/usr/lib/glibc
```

The meaning of the new configure options:

`--libexecdir=/usr/lib/glibc`
This changes the location of the **pt_chown** program from its default of `/usr/libexec` to `/usr/lib/glibc`.

Compile the package:

```
make
```

Важно

In this section, the test suite for Glibc is considered critical. Do not skip it under any circumstance.

Before running the tests, copy a file from the source tree into our build tree to prevent a couple of test failures, then test the results:

```
cp -v ../glibc-2.13/iconvdata/gconv-modules iconvdata
make -k check 2>&1 | tee glibc-check-log
grep Error glibc-check-log
```

You will probably see an expected (ignored) failure in the *posix/annexc* test. In addition the Glibc test suite is somewhat dependent on the host system. This is a list of the most common issues:

- The *nptl/tst-clock2*, *nptl/tst-attr3*, and *rt/tst-cpuclock2* tests have been known to fail. The reason is not completely understood, but indications are that minor timing issues can trigger these failures.
- The math tests sometimes fail when running on systems where the CPU is not a relatively new genuine Intel or authentic AMD processor.

- If you have mounted the LFS partition with the *noatime* option, the *atime* test will fail. As mentioned in Раздел 2.4, «Монтирование нового раздела», do not use the *noatime* option while building LFS.
- When running on older and slower hardware or on systems under load, some tests can fail because of test timeouts being exceeded. Modifying the make check command to set a TIMEOUTFACTOR is reported to help eliminate these errors (e.g. **TIMEOUTFACTOR=16 make -k check**).

Though it is a harmless message, the install stage of Glibc will complain about the absence of `/etc/ld.so.conf`. Prevent this warning with:

```
touch /etc/ld.so.conf
```

Install the package:

```
make install
```

The locales that can make the system respond in a different language were not installed by the above command. None of the locales are required, but if some of them are missing, test suites of the future packages would skip important testcases.

Individual locales can be installed using the **localedef** program. E.g., the first **localedef** command below combines the `/usr/share/i18n/locales/cs_CZ` charset-independent locale definition with the `/usr/share/i18n/charmaps/UTF-8.gz` charmap definition and appends the result to the `/usr/lib/locale/locale-archive` file. The following instructions will install the minimum set of locales necessary for the optimal coverage of tests:

```
mkdir -pv /usr/lib/locale
localedef -i cs_CZ -f UTF-8 cs_CZ.UTF-8
localedef -i de_DE -f ISO-8859-1 de_DE
localedef -i de_DE@euro -f ISO-8859-15 de_DE@euro
localedef -i de_DE -f UTF-8 de_DE.UTF-8
localedef -i en_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 ja_JP -f EUC-JP ja_JP
localedef -i tr_TR -f UTF-8 tr_TR.UTF-8
localedef -i zh_CN -f GB18030 zh_CN.GB18030
```

In addition, install the locale for your own country, language and character set.

Alternatively, install all locales listed in the `glibc-2.13/localedata/SUPPORTED` file (it includes every locale listed above and many more) at once with the following time-consuming command:

```
make localedata/install-locales
```

Then use the **localedef** command to create and install locales not listed in the `glibc-2.13/localedata/SUPPORTED` file in the unlikely case you need them.

6.9.2. Configuring Glibc

The `/etc/nsswitch.conf` file needs to be created because, although Glibc provides defaults when this file is missing or corrupt, the Glibc defaults do not work well in a networked environment. The time zone also needs to be configured.

Create a new file `/etc/nsswitch.conf` by running the following:

```
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
```

One way to determine the local time zone, run the following script:

```
tzselect
```

After answering a few questions about the location, the script will output the name of the time zone (e.g., *America/Edmonton*). There are also some other possible timezones listed in `/usr/share/zoneinfo` such as *Canada/Eastern* or *EST5EDT* that are not identified by the script but can be used.

Then create the `/etc/localtime` file by running:

```
cp -v --remove-destination /usr/share/zoneinfo/<xxx> \
    /etc/localtime
```

Replace <*xxx*> with the name of the time zone selected (e.g., Canada/Eastern).

The meaning of the cp option:

`--remove-destination`
: This is needed to force removal of the already existing symbolic link. The reason for copying the file instead of using a symlink is to cover the situation where `/usr` is on a separate partition. This could be important when booted into single user mode.

6.9.3. Configuring the Dynamic Loader

By default, the dynamic loader (`/lib/ld-linux.so.2`) searches through `/lib` and `/usr/lib` for dynamic libraries that are needed by programs as they are run. However, if there are libraries in directories other than `/lib` and `/usr/lib`, these need to be added to the

`/etc/ld.so.conf` file in order for the dynamic loader to find them. Two directories that are commonly known to contain additional libraries are `/usr/local/lib` and `/opt/lib`, so add those directories to the dynamic loader's search path.

Create a new file `/etc/ld.so.conf` by running the following:

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

/usr/local/lib
/opt/lib

# End /etc/ld.so.conf
EOF
```

6.9.4. Contents of Glibc

Installed programs:	catchsegv, gencat, getconf, getent, iconv, iconvconfig, ldconfig, ldd, lddlibc4, locale, localedef, mtrace, nscd, pcprofiledump, pt_chown, rpcgen, rpcinfo, sln, sprof, tzselect, xtrace, zdump, and zic
Installed libraries:	ld.so, libBrokenLocale.{a,so}, libSegFault.so, libanl.{a,so}, libbsd-compat.a, libc.{a,so}, libc_nonshared.a, libcidn.so, libcrypt.{a,so}, libdl.{a,so}, libg.a, libieee.a, libm.{a,so}, libmcheck.a, libmemusage.so, libnsl.{a,so}, libnss_compat.so, libnss_dns.so, libnss_files.so, libnss_hesiod.so, libnss_nis.so, libnss_nisplus.so, libpcprofile.so, libpthread.{a,so}, libpthread_nonshared.a, libresolv.{a,so}, librpcsvc.a, librt.{a,so}, libthread_db.so, and libutil.{a,so}
Installed directories:	/usr/include/arpa, /usr/include/bits, /usr/include/gnu, /usr/include/net, /usr/include/netash, /usr/include/netatalk, /usr/include/netax25, /usr/include/neteconet, /usr/include/netinet, /usr/include/netipx, /usr/include/netiucv, /usr/include/netpacket, /usr/include/netrom, /usr/include/netrose, /usr/include/nfs, /usr/include/protocols, /usr/include/rpc, /usr/include/rpcsvc, /usr/include/sys, /usr/lib/gconv, /usr/lib/glibc, /usr/lib/locale, /usr/share/i18n, /usr/share/zoneinfo

Short Descriptions

catchsegv	Can be used to create a stack trace when a program terminates with a segmentation fault
gencat	Generates message catalogues
getconf	Displays the system configuration values for file system specific variables
getent	Gets entries from an administrative database
iconv	Performs character set conversion
iconvconfig	Creates fastloading **iconv** module configuration files
ldconfig	Configures the dynamic linker runtime bindings
ldd	Reports which shared libraries are required by each given program or shared library

lddlibc4	Assists **ldd** with object files
locale	Prints various information about the current locale
localedef	Compiles locale specifications
mtrace	Reads and interprets a memory trace file and displays a summary in human-readable format
nscd	A daemon that provides a cache for the most common name service requests
pcprofiledump	Dumps information generated by PC profiling
pt_chown	A helper program for **grantpt** to set the owner, group and access permissions of a slave pseudo terminal
rpcgen	Generates C code to implement the Remote Procedure Call (RPC) protocol
rpcinfo	Makes an RPC call to an RPC server
sln	A statically linked **ln** program
sprof	Reads and displays shared object profiling data
tzselect	Asks the user about the location of the system and reports the corresponding time zone description
xtrace	Traces the execution of a program by printing the currently executed function
zdump	The time zone dumper
zic	The time zone compiler
`ld.so`	The helper program for shared library executables
`libBrokenLocale`	Used internally by Glibc as a gross hack to get broken programs (e.g., some Motif applications) running. See comments in `glibc-2.13/locale/broken_cur_max.c` for more information
`libSegFault`	The segmentation fault signal handler, used by **catchsegv**
`libanl`	An asynchronous name lookup library
`libbsd-compat`	Provides the portability needed in order to run certain Berkeley Software Distribution (BSD) programs under Linux
`libc`	The main C library
`libcidn`	Used internally by Glibc for handling internationalized domain names in the `getaddrinfo()` function
`libcrypt`	The cryptography library
`libdl`	The dynamic linking interface library
`libg`	Dummy library containing no functions. Previously was a runtime library for **g++**
`libieee`	Linking in this module forces error handling rules for math functions as defined by the Institute of Electrical and Electronic Engineers (IEEE). The default is POSIX.1 error handling
`libm`	The mathematical library
`libmcheck`	Turns on memory allocation checking when linked to

`libmemusage`	Used by **memusage** to help collect information about the memory usage of a program
`libnsl`	The network services library
`libnss`	The Name Service Switch libraries, containing functions for resolving host names, user names, group names, aliases, services, protocols, etc.
`libpcprofile`	Contains profiling functions used to track the amount of CPU time spent in specific source code lines
`libpthread`	The POSIX threads library
`libresolv`	Contains functions for creating, sending, and interpreting packets to the Internet domain name servers
`librpcsvc`	Contains functions providing miscellaneous RPC services
`librt`	Contains functions providing most of the interfaces specified by the POSIX.1b Realtime Extension
`libthread_db`	Contains functions useful for building debuggers for multi-threaded programs
`libutil`	Contains code for «standard» functions used in many different Unix utilities

6.10. Re-adjusting the Toolchain

Now that the final C libraries have been installed, it is time to adjust the toolchain again. The toolchain will be adjusted so that it will link any newly compiled program against these new libraries. This is a similar process used in the «Adjusting» phase in the beginning of Глава 5, but with the adjustments reversed. In Глава 5, the chain was guided from the host's `/{,usr/}lib` directories to the new `/tools/lib` directory. Now, the chain will be guided from that same `/tools/lib` directory to the LFS `/{,usr/}lib` directories.

First, backup the `/tools` linker, and replace it with the adjusted linker we made in chapter 5. We'll also create a link to its counterpart in `/tools/$(gcc -dumpmachine)/bin`:

```
mv -v /tools/bin/{ld,ld-old}
mv -v /tools/$(gcc -dumpmachine)/bin/{ld,ld-old}
mv -v /tools/bin/{ld-new,ld}
ln -sv /tools/bin/ld /tools/$(gcc -dumpmachine)/bin/ld
```

Next, amend the GCC specs file so that it points to the new dynamic linker. Simply deleting all instances of «/tools» should leave us with the correct path to the dynamic linker. Also adjust the specs file so that GCC knows where to find the correct headers and Glibc start files. A **sed** command accomplishes this:

```
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
```

It is a good idea to visually inspect the specs file to verify the intended change was actually made.

It is imperative at this point to ensure that the basic functions (compiling and linking) of the adjusted toolchain are working as expected. To do this, perform the following sanity checks:

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

If everything is working correctly, there should be no errors, and the output of the last command will be (allowing for platform-specific differences in dynamic linker name):

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

Note that `/lib` is now the prefix of our dynamic linker.

Now make sure that we're setup to use the correct startfiles:

```
grep -o '/usr/lib.*/crt[1in].*succeeded' dummy.log
```

If everything is working correctly, there should be no errors, and the output of the last command will be:

```
/usr/lib/crt1.o succeeded
/usr/lib/crti.o succeeded
/usr/lib/crtn.o succeeded
```

Verify that the compiler is searching for the correct header files:

```
grep -B1 '^ /usr/include' dummy.log
```

This command should return successfully with the following output:

```
#include <...> search starts here:
 /usr/include
```

Next, verify that the new linker is being used with the correct search paths:

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

If everything is working correctly, there should be no errors, and the output of the last command (allowing for platform-specific target triplets) will be:

```
SEARCH_DIR("/tools/i686-pc-linux-gnu/lib")
SEARCH_DIR("/usr/lib")
SEARCH_DIR("/lib");
```

Next make sure that we're using the correct libc:

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

If everything is working correctly, there should be no errors, and the output of the last command (allowing for a lib64 directory on 64-bit hosts) will be:

```
attempt to open /lib/libc.so.6 succeeded
```

Lastly, make sure GCC is using the correct dynamic linker:

```
grep found dummy.log
```

If everything is working correctly, there should be no errors, and the output of the last command will be (allowing for platform-specific differences in dynamic linker name and a lib64 directory on 64-bit hosts):

```
found ld-linux.so.2 at /lib/ld-linux.so.2
```

If the output does not appear as shown above or is not received at all, then something is seriously wrong. Investigate and retrace the steps to find out where the problem is and correct it. The most likely reason is that something went wrong with the specs file adjustment. Any issues will need to be resolved before continuing on with the process.

Once everything is working correctly, clean up the test files:

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

6.11. Zlib-1.2.5

The Zlib package contains compression and decompression routines used by some programs.

Приблизительное время сборки: less than 0.1 SBU
Требует свободного места на диске: 2.8 MB

6.11.1. Installation of Zlib

First, fix a typo in the package header file:

```
sed -i 's/ifdef _LARGEFILE64_SOURCE/ifndef _LARGEFILE64_SOURCE/' zlib.h
```

Prepare Zlib for compilation:

```
CFLAGS='-mstackrealign -fPIC -O3' ./configure --prefix=/usr
```

The meaning of the new configure environment variable:

`CFLAGS='-mstackrealign -fPIC -O3'`
Setting CFLAGS overrides the default optimization in the package to prevent some run time errors. Note that the -mstackrealign may cause build failures in non-Intel architecture systems.

Compile the package:

```
make
```

To test the results, issue:

```
make check
```

Install the package:

```
make install
```

The shared library needs to be moved to `/lib`, and as a result the `.so` file in `/usr/lib` will need to be recreated:

```
mv -v /usr/lib/libz.so.* /lib
ln -sfv ../../lib/libz.so.1.2.5 /usr/lib/libz.so
```

6.11.2. Contents of Zlib

Installed libraries: libz.{a,so}

Short Descriptions

`libz` Contains compression and decompression functions used by some programs

6.12. Binutils-2.21

The Binutils package contains a linker, an assembler, and other tools for handling object files.

Приблизительное время сборки:	2.1 SBU
Требует свободного места на диске:	222 MB

6.12.1. Installation of Binutils

Verify that the PTYs are working properly inside the chroot environment by performing a simple test:

```
expect -c "spawn ls"
```

This command should output the following:

```
spawn ls
```

If, instead, the output includes the message below, then the environment is not set up for proper PTY operation. This issue needs to be resolved before running the test suites for Binutils and GCC:

```
The system has no more ptys.
Ask your system administrator to create more.
```

Suppress the installation of an outdated `standards.info` file as a newer one is installed later on in the Autoconf instructions:

```
rm -fv etc/standards.info
sed -i.bak '/^INFO/s/standards.info //' etc/Makefile.in
```

The Binutils documentation recommends building Binutils outside of the source directory in a dedicated build directory:

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

Prepare Binutils for compilation:

```
../binutils-2.21/configure --prefix=/usr \
    --enable-shared
```

Compile the package:

```
make tooldir=/usr
```

The meaning of the make parameter:

`tooldir=/usr`
Normally, the tooldir (the directory where the executables will ultimately be located) is set to `$(exec_prefix)/$(target_alias)`. For example, x86_64 machines would expand that to `/usr/x86_64-unknown-linux-gnu`. Because this is a custom system, this target-specific directory in `/usr` is not required. `$(exec_prefix)/$(target_`

`alias`) would be used if the system was used to cross-compile (for example, compiling a package on an Intel machine that generates code that can be executed on PowerPC machines).

Важно

The test suite for Binutils in this section is considered critical. Do not skip it under any circumstances.

Test the results:

```
make check
```

Install the package:

```
make tooldir=/usr install
```

Install the `libiberty` header file that is needed by some packages:

```
cp -v ../binutils-2.21/include/libiberty.h /usr/include
```

6.12.2. Contents of Binutils

Installed programs:	addr2line, ar, as, c++filt, gprof, ld, nm, objcopy, objdump, ranlib, readelf, size, strings, and strip
Installed libraries:	libiberty.a, libbfd.{a,so}, and libopcodes.{a,so}
Installed directory:	/usr/lib/ldscripts

Short Descriptions

addr2line	Translates program addresses to file names and line numbers; given an address and the name of an executable, it uses the debugging information in the executable to determine which source file and line number are associated with the address
ar	Creates, modifies, and extracts from archives
as	An assembler that assembles the output of **gcc** into object files
c++filt	Used by the linker to de-mangle C++ and Java symbols and to keep overloaded functions from clashing
gprof	Displays call graph profile data
ld	A linker that combines a number of object and archive files into a single file, relocating their data and tying up symbol references
nm	Lists the symbols occurring in a given object file
objcopy	Translates one type of object file into another
objdump	Displays information about the given object file, with options controlling the particular information to display; the information shown is useful to programmers who are working on the compilation tools
ranlib	Generates an index of the contents of an archive and stores it in the archive; the index lists all of the symbols defined by archive members that are relocatable object files
readelf	Displays information about ELF type binaries
size	Lists the section sizes and the total size for the given object files

strings	Outputs, for each given file, the sequences of printable characters that are of at least the specified length (defaulting to four); for object files, it prints, by default, only the strings from the initializing and loading sections while for other types of files, it scans the entire file
strip	Discards symbols from object files
`libiberty`	Contains routines used by various GNU programs, including **getopt**, **obstack**, **strerror**, **strtol**, and **strtoul**
`libbfd`	The Binary File Descriptor library
`libopcodes`	A library for dealing with opcodes—the «readable text» versions of instructions for the processor; it is used for building utilities like **objdump**.

6.13. GMP-5.0.1

The GMP package contains math libraries. These have useful functions for arbitrary precision arithmetic.

Приблизительное время сборки:	1.7 SBU
Требует свободного места на диске:	39 MB

6.13.1. Installation of GMP

Замечание

If you are building for 32-bit x86, but you have a CPU which is capable of running 64-bit code *and* you have specified `CFLAGS` in the environment, the configure script will attempt to configure for 64-bits and fail. Avoid this by invoking the configure command below with

```
ABI=32 ./configure ...
```

Prepare GMP for compilation:

```
./configure --prefix=/usr --enable-cxx --enable-mpbsd
```

The meaning of the new configure options:

`--enable-cxx`
This parameter enables C++ support

`--enable-mpbsd`
This builds the Berkeley MP compatibility library

Compile the package:

```
make
```

Важно

The test suite for GMP in this section is considered critical. Do not skip it under any circumstances.

Test the results:

```
make check 2>&1 | tee gmp-check-log
```

Ensure that all 162 tests in the test suite passed. Check the results by issuing the following command:

```
awk '/tests passed/{total+=$2} ; END{print total}' gmp-check-log
```

Install the package:

```
make install
```

If desired, install the documentation:

```
mkdir -v /usr/share/doc/gmp-5.0.1
cp    -v doc/{isa_abi_headache,configuration} doc/*.html \
         /usr/share/doc/gmp-5.0.1
```

6.13.2. Contents of GMP

Installed Libraries: libgmp.{a,so}, libgmpxx.{a,so}, and libmp.{a,so}
Installed directory: /usr/share/doc/gmp-5.0.1

Short Descriptions

`libgmp`	Contains precision math functions.
`libgmpxx`	Contains C++ precision math functions.
`libmp`	Contains the Berkeley MP math functions.

6.14. MPFR-3.0.0

The MPFR package contains functions for multiple precision math.

Приблизительное время сборки:	1.1 SBU
Требует свободного места на диске:	27.1 MB

6.14.1. Installation of MPFR

Prepare MPFR for compilation:

```
./configure --prefix=/usr --enable-thread-safe \
  --docdir=/usr/share/doc/mpfr-3.0.0
```

Compile the package:

```
make
```

Важно

The test suite for MPFR in this section is considered critical. Do not skip it under any circumstances.

Test the results and ensure that all tests passed:

```
make check
```

Install the package:

```
make install
```

Install the documentation:

```
make html
make install-html
```

6.14.2. Contents of MPFR

Installed Libraries:	libmpfr.{a,so}
Installed directory:	/usr/share/doc/mpfr-3.0.0

Short Descriptions

`libmpfr` Contains multiple-precision math functions.

6.15. MPC-0.8.2

The MPC package contains a library for the arithmetic of complex numbers with arbitrarily high precision and correct rounding of the result.

Приблизительное время сборки: 0.3 SBU
Требует свободного места на диске: 10.5 MB

6.15.1. Installation of MPC

Prepare MPC for compilation:

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

Compile the package:

```
make
```

To test the results, issue:

```
make check
```

Install the package:

```
make install
```

6.15.2. Contents of MPC

Installed Libraries: libmpc.{a,so}

Short Descriptions

`libmpc` Contains complex math functions

6.16. GCC-4.5.2

The GCC package contains the GNU compiler collection, which includes the C and C++ compilers.

Приблизительное время сборки:	44 SBU
Требует свободного места на диске:	1.1 GB

6.16.1. Installation of GCC

Apply a **sed** substitution that will suppress the installation of `libiberty.a`. The version of `libiberty.a` provided by Binutils will be used instead:

```
sed -i 's/install_to_$(INSTALL_DEST) //' libiberty/Makefile.in
```

As in Раздел 5.10, «GCC-4.5.2 - Шаг 2», apply the following **sed** to force the build to use the `-fomit-frame-pointer` compiler flag in order to ensure consistent compiler builds:

```
case `uname -m` in
  i?86) sed -i 's/^T_CFLAGS =$/& -fomit-frame-pointer/' \
        gcc/Makefile.in ;;
esac
```

The **fixincludes** script is known to occasionally erroneously attempt to "fix" the system headers installed so far. As the headers up to this point are known to not require fixing, issue the following command to prevent the **fixincludes** script from running:

```
sed -i 's@\./fixinc\.sh@-c true@' gcc/Makefile.in
```

The GCC documentation recommends building GCC outside of the source directory in a dedicated build directory:

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

Prepare GCC for compilation:

```
../gcc-4.5.2/configure --prefix=/usr \
    --libexecdir=/usr/lib --enable-shared \
    --enable-threads=posix --enable-__cxa_atexit \
    --enable-clocale=gnu --enable-languages=c,c++ \
    --disable-multilib --disable-bootstrap --with-system-zlib
```

Note that for other languages, there are some prerequisites that are not available. See the BLFS Book for instructions on how to build all the GCC supported languages.

The meaning of the new configure option:

`--with-system-zlib`
: This switch tells GCC to link to the system installed copy of the Zlib library, rather than its own internal copy.

Compile the package:

```
make
```

Важно

In this section, the test suite for GCC is considered critical. Do not skip it under any circumstance.

One set of tests in the GCC test suite is known to exhaust the stack, so increase the stack size prior to running the tests:

```
ulimit -s 16384
```

Test the results, but do not stop at errors:

```
make -k check
```

To receive a summary of the test suite results, run:

```
../gcc-4.5.2/contrib/test_summary
```

For only the summaries, pipe the output through **grep -A7 Summ**.

Results can be compared with those located at *http://www.linuxfromscratch.org/lfs/build-logs/6.8/* and *http://gcc.gnu.org/ml/gcc-testresults/*.

A few unexpected failures cannot always be avoided. The GCC developers are usually aware of these issues, but have not resolved them yet. In particular, the `libmudflap` tests are known be particularly problematic as a result of a bug in GCC (*http://gcc.gnu.org/bugzilla/show_bug.cgi?id=20003*). Unless the test results are vastly different from those at the above URL, it is safe to continue.

Install the package:

```
make install
```

Some packages expect the C preprocessor to be installed in the `/lib` directory. To support those packages, create this symlink:

```
ln -sv ../usr/bin/cpp /lib
```

Many packages use the name **cc** to call the C compiler. To satisfy those packages, create a symlink:

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

Now that our final toolchain is in place, it is important to again ensure that compiling and linking will work as expected. We do this by performing the same sanity checks as we did earlier in the chapter:

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

If everything is working correctly, there should be no errors, and the output of the last command will be (allowing for platform-specific differences in dynamic linker name):

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

Now make sure that we're setup to use the correct startfiles:

```
grep -o '/usr/lib.*/crt[1in].*succeeded' dummy.log
```

If everything is working correctly, there should be no errors, and the output of the last command will be:

```
/usr/lib/gcc/i686-pc-linux-gnu/4.5.2/../../../crt1.o succeeded
/usr/lib/gcc/i686-pc-linux-gnu/4.5.2/../../../crti.o succeeded
/usr/lib/gcc/i686-pc-linux-gnu/4.5.2/../../../crtn.o succeeded
```

Depending on your machine architecture, the above may differ slightly, the difference usually being the name of the directory after `/usr/lib/gcc`. If your machine is a 64-bit system, you may also see a directory named `lib64` towards the end of the string. The important thing to look for here is that **gcc** has found all three `crt*.o` files under the `/usr/lib` directory.

Verify that the compiler is searching for the correct header files:

```
grep -B4 '^ /usr/include' dummy.log
```

This command should return successfully with the following output:

```
#include <...> search starts here:
 /usr/local/include
 /usr/lib/gcc/i686-pc-linux-gnu/4.5.2/include
 /usr/lib/gcc/i686-pc-linux-gnu/4.5.2/include-fixed
 /usr/include
```

Again, note that the directory named after your target triplet may be different than the above, depending on your architecture.

Замечание

As of version 4.3.0, GCC now unconditionally installs the `limits.h` file into the private `include-fixed` directory, and that directory is required to be in place.

Next, verify that the new linker is being used with the correct search paths:

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

If everything is working correctly, there should be no errors, and the output of the last command (allowing for platform-specific target triplets) will be:

```
SEARCH_DIR("/usr/i686-pc-linux-gnu/lib")
SEARCH_DIR("/usr/local/lib")
SEARCH_DIR("/lib")
SEARCH_DIR("/usr/lib");
```

A 64-bit system may see a few more directories. For example, here is the output from an x86_64 machine:

```
SEARCH_DIR("/usr/x86_64-unknown-linux-gnu/lib64")
SEARCH_DIR("/usr/local/lib64")
SEARCH_DIR("/lib64")
SEARCH_DIR("/usr/lib64")
SEARCH_DIR("/usr/x86_64-unknown-linux-gnu/lib")
SEARCH_DIR("/usr/local/lib")
SEARCH_DIR("/lib")
SEARCH_DIR("/usr/lib");
```

Next make sure that we're using the correct libc:

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

If everything is working correctly, there should be no errors, and the output of the last command (allowing for a lib64 directory on 64-bit hosts) will be:

```
attempt to open /lib/libc.so.6 succeeded
```

Lastly, make sure GCC is using the correct dynamic linker:

```
grep found dummy.log
```

If everything is working correctly, there should be no errors, and the output of the last command will be (allowing for platform-specific differences in dynamic linker name and a lib64 directory on 64-bit hosts):

```
found ld-linux.so.2 at /lib/ld-linux.so.2
```

If the output does not appear as shown above or is not received at all, then something is seriously wrong. Investigate and retrace the steps to find out where the problem is and correct it. The most likely reason is that something went wrong with the specs file adjustment. Any issues will need to be resolved before continuing on with the process.

Once everything is working correctly, clean up the test files:

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

6.16.2. Contents of GCC

Installed programs:	c++, cc (link to gcc), cpp, g++, gcc, gccbug, and gcov
Installed libraries:	libgcc.a, libgcc_eh.a, libgcc_s.so, libgcov.a, libgomp.{a,so}, libmudflap.{a,so}, libmudflapth.{a,so}, libssp.{a,so}, libssp_nonshared.a, libstdc++.{a,so} and libsupc++.a
Installed directories:	/usr/include/c++, /usr/lib/gcc, /usr/share/gcc-4.5.2

Short Descriptions

c++	The C++ compiler
cc	The C compiler
cpp	The C preprocessor; it is used by the compiler to expand the #include, #define, and similar statements in the source files
g++	The C++ compiler
gcc	The C compiler
gccbug	A shell script used to help create useful bug reports
gcov	A coverage testing tool; it is used to analyze programs to determine where optimizations will have the most effect
`libgcc`	Contains run-time support for **gcc**
`libgcov`	This library is linked in to a program when GCC is instructed to enable profiling

`libgomp`	GNU implementation of the OpenMP API for multi-platform shared-memory parallel programming in C/C++ and Fortran
`libmudflap`	Contains routines that support GCC's bounds checking functionality
`libssp`	Contains routines supporting GCC's stack-smashing protection functionality
`libstdc++`	The standard C++ library
`libsupc++`	Provides supporting routines for the C++ programming language

6.17. Sed-4.2.1

The Sed package contains a stream editor.

Приблизительное время сборки:	0.2 SBU
Требует свободного места на диске:	8.3 MB

6.17.1. Installation of Sed

Prepare Sed for compilation:

```
./configure --prefix=/usr --bindir=/bin --htmldir=/usr/share/doc/sed-4.2.1
```

The meaning of the new configure option:

`--htmldir`
This sets the directory where the HTML documentation will be installed to.

Compile the package:

```
make
```

Generate the HTML documentation:

```
make html
```

To test the results, issue:

```
make check
```

Install the package:

```
make install
```

Install the HTML documentation:

```
make -C doc install-html
```

6.17.2. Contents of Sed

Installed program:	sed
Installed directory:	/usr/share/doc/sed-4.2.1

Short Descriptions

sed Filters and transforms text files in a single pass

6.18. Pkg-config-0.25

The pkg-config package contains a tool for passing the include path and/or library paths to build tools during the configure and make file execution.

Приблизительное время сборки:	0.3 SBU
Требует свободного места на диске:	11.5 MB

6.18.1. Installation of Pkg-config

Замечание

Pkg-Config will use an included version of Popt to parse command line options. If an external version of Popt is desired, install that version using the *BLFS Popt build instructions* before installing Pkg-config.

Prepare Pkg-config for compilation:

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

Compile the package:

```
make
```

To test the results, issue:

```
make check
```

Install the package:

```
make install
```

6.18.2. Contents of Pkg-config

Installed program: pkg-config

Short Descriptions

pkg-config	Returns meta information for the specified library or package.

6.19. Ncurses-5.7

The Ncurses package contains libraries for terminal-independent handling of character screens.

Приблизительное время сборки:	0.8 SBU
Требует свободного места на диске:	35 MB

6.19.1. Installation of Ncurses

Prepare Ncurses for compilation:

```
./configure --prefix=/usr --with-shared --without-debug --enable-widec
```

The meaning of the configure option:

`--enable-widec`
This switch causes wide-character libraries (e.g., `libncursesw.so.5.7`) to be built instead of normal ones (e.g., `libncurses.so.5.7`). These wide-character libraries are usable in both multibyte and traditional 8-bit locales, while normal libraries work properly only in 8-bit locales. Wide-character and normal libraries are source-compatible, but not binary-compatible.

Compile the package:

```
make
```

This package has a test suite, but it can only be run after the package has been installed. The tests reside in the `test/` directory. See the `README` file in that directory for further details.

Install the package:

```
make install
```

Move the shared libraries to the `/lib` directory, where they are expected to reside:

```
mv -v /usr/lib/libncursesw.so.5* /lib
```

Because the libraries have been moved, one symlink points to a non-existent file. Recreate it:

```
ln -sfv ../../lib/libncursesw.so.5 /usr/lib/libncursesw.so
```

Many applications still expect the linker to be able to find non-wide-character Ncurses libraries. Trick such applications into linking with wide-character libraries by means of symlinks and linker scripts:

```
for lib in ncurses form panel menu ; do \
    rm -vf /usr/lib/lib${lib}.so ; \
    echo "INPUT(-l${lib}w)" >/usr/lib/lib${lib}.so ; \
    ln -sfv lib${lib}w.a /usr/lib/lib${lib}.a ; \
done
ln -sfv libncurses++w.a /usr/lib/libncurses++.a
```

Finally, make sure that old applications that look for `-lcurses` at build time are still buildable:

```
rm -vf /usr/lib/libcursesw.so
echo "INPUT(-lncursesw)" >/usr/lib/libcursesw.so
ln -sfv libncurses.so /usr/lib/libcurses.so
ln -sfv libncursesw.a /usr/lib/libcursesw.a
ln -sfv libncurses.a /usr/lib/libcurses.a
```

If desired, install the Ncurses documentation:

```
mkdir -v       /usr/share/doc/ncurses-5.7
cp -v -R doc/* /usr/share/doc/ncurses-5.7
```

Замечание

The instructions above don't create non-wide-character Ncurses libraries since no package installed by compiling from sources would link against them at runtime. If you must have such libraries because of some binary-only application or to be compliant with LSB, build the package again with the following commands:

```
make distclean
./configure --prefix=/usr --with-shared --without-normal \
  --without-debug --without-cxx-binding
make sources libs
cp -av lib/lib*.so.5* /usr/lib
```

6.19.2. Contents of Ncurses

Installed programs:	captoinfo (link to tic), clear, infocmp, infotocap (link to tic), ncursesw5-config, reset (link to tset), tic, toe, tput, and tset
Installed libraries:	libcursesw.{a,so} (symlink and linker script to libncursesw.{a,so}), libformw.{a,so}, libmenuw.{a,so}, libncurses++w.a, libncursesw.{a,so}, libpanelw.{a,so} and their non-wide-character counterparts without "w" in the library names.
Installed directories:	/usr/share/tabset, /usr/share/terminfo

Short Descriptions

captoinfo	Converts a termcap description into a terminfo description
clear	Clears the screen, if possible
infocmp	Compares or prints out terminfo descriptions
infotocap	Converts a terminfo description into a termcap description
ncursesw5-config	Provides configuration information for ncurses
reset	Reinitializes a terminal to its default values
tic	The terminfo entry-description compiler that translates a terminfo file from source format into the binary format needed for the ncurses library routines. A terminfo file contains information on the capabilities of a certain terminal

toe	Lists all available terminal types, giving the primary name and description for each
tput	Makes the values of terminal-dependent capabilities available to the shell; it can also be used to reset or initialize a terminal or report its long name
tset	Can be used to initialize terminals
`libcurses`	A link to `libncurses`
`libncurses`	Contains functions to display text in many complex ways on a terminal screen; a good example of the use of these functions is the menu displayed during the kernel's **make menuconfig**
`libform`	Contains functions to implement forms
`libmenu`	Contains functions to implement menus
`libpanel`	Contains functions to implement panels

6.20. Util-linux-2.19

The Util-linux package contains miscellaneous utility programs. Among them are utilities for handling file systems, consoles, partitions, and messages.

Приблизительное время сборки:	0.6 SBU
Требует свободного места на диске:	50 MB

6.20.1. FHS compliance notes

The FHS recommends using the `/var/lib/hwclock` directory instead of the usual `/etc` directory as the location for the `adjtime` file. To make the **hwclock** program FHS-compliant, run the following:

```
sed -e 's@etc/adjtime@var/lib/hwclock/adjtime@g' \
    -i $(grep -rl '/etc/adjtime' .)
mkdir -pv /var/lib/hwclock
```

6.20.2. Installation of Util-linux

```
./configure --enable-arch --enable-partx --enable-write
```

The meaning of the configure options:

`--enable-arch`
Enables building the **arch** program

`--enable-partx`
Enables building the **addpart**, **delpart** and **partx** programs

`--enable-write`
Enables building the **write** program

Compile the package:

```
make
```

This package does not come with a test suite.

Install the package:

```
make install
```

6.20.3. Contents of Util-linux

Installed programs:	addpart, agetty, arch, blkid, blockdev, cal, cfdisk, chkdupexe, chrt, col, colcrt, colrm, column, ctrlaltdel, cytune, ddate, delpart, dmesg, fallocate, fdformat, fdisk, findfs, findmnt, flock, fsck, fsck.cramfs, fsck.minix, fsfreeze, fstrim, getopt, hexdump, hwclock, i386, ionice, ipcmk, ipcrm, ipcs, isosize, ldattach, line, linux32, linux64, logger, look, losetup, lsblk, lscpu, mcookie, mkfs, mkfs.bfs, mkfs.cramfs, mkfs.minix, mkswap, more, mount, namei, partx, pg, pivot_root, readprofile, rename, renice, rev, rtcwake, script, scriptreplay, setarch, setsid, setterm, sfdisk, swaplabel, swapoff (link to swapon), swapon, switch_root, tailf, taskset, tunelp, ul, umount, unshare, uuidd, uuidgen, wall, whereis, wipefs, and write
Installed libraries:	libblkid.{a,so}, libmount.{a,so}, libuuid.{a,so}
Installed directories:	/usr/share/getopt, /var/lib/hwclock

Short Descriptions

addpart	Informs the Linux kernel of new partitions
agetty	Opens a tty port, prompts for a login name, and then invokes the **login** program
arch	Reports the machine's architecture
blkid	A command line utility to locate and print block device attributes
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
chkdupexe	Finds duplicate executables
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
cytune	Tunes the parameters of the serial line drivers for Cyclades cards
ddate	Gives the Discordian date or converts the given Gregorian date to a Discordian one
delpart	Asks the Linux kernel to remove a partition
dmesg	Dumps the kernel boot messages
fallocate	Preallocates space to a file
fdformat	Low-level formats a floppy disk
fdisk	Manipulates the paritition table of the given device

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

ldattach Attaches a line discipline to a serial line

line Copies a single 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 architechture information

mcookie Generates magic cookies (128-bit random hexadecimal numbers) for **xauth**

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

namei	Shows the symbolic links in the given pathnames
partx	Tells the kernel about the presence and numbering of on-disk partitions
pg	Displays a text file one screen full at a time
pivot_root	Makes the given file system the new root file system of the current process
readprofile	Reads kernel profiling information
rename	Renames the given files, replacing a given string with another
renice	Alters the priority of running processes
rev	Reverses the lines of a given file
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
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
tunelp	Tunes the parameters of the line printer
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
unshare	Runs a program with some namespaces unshared from parent
uuidd	A daemon used by the UUID library to generate time-based UUIDs in a secure and guranteed-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
wall	Displays the contents of a file or, by default, its standard input, on the terminals of all currently logged in users
whereis	Reports the location of the binary, source, and man page for the given command
wipefs	Wipes a filesystem signature from a device
write	Sends a message to the given user *if* that user has not disabled receipt of such messages

`libblkid`	Contains routines for device identification and token extraction
`libuuid`	Contains routines for generating unique identifiers for objects that may be accessible beyond the local system

6.21. E2fsprogs-1.41.14

The E2fsprogs package contains the utilities for handling the ext2 file system. It also supports the ext3 and ext4 journaling file systems.

Приблизительное время сборки: 0.5 SBU
Требует свободного места на диске: 45 MB

6.21.1. Installation of E2fsprogs

The E2fsprogs documentation recommends that the package be built in a subdirectory of the source tree:

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

Prepare E2fsprogs for compilation:

```
../configure --prefix=/usr --with-root-prefix="" \
    --enable-elf-shlibs --disable-libblkid --disable-libuuid \
    --disable-uuidd --disable-fsck
```

The meaning of the configure options:

--with-root-prefix=""
Certain programs (such as the **e2fsck** program) are considered essential programs. When, for example, /usr is not mounted, these programs still need to be available. They belong in directories like /lib and /sbin. If this option is not passed to E2fsprogs' configure, the programs are installed into the /usr directory.

--enable-elf-shlibs
This creates the shared libraries which some programs in this package use.

*--disable-**
This prevents E2fsprogs from building and installing the libuuid and libblkid libraries, the uuidd daemon, and the **fsck** wrapper, as Util-Linux installed all of them earlier.

Compile the package:

```
make
```

To test the results, issue:

```
make check
```

One of the E2fsprogs tests will attempt to allocate 256 MB of memory. If you do not have significantly more RAM than this, it is recommended to enable sufficient swap space for the test. See Раздел 2.3, «Создание файловой системы на разделе» and Раздел 2.4, «Монтирование нового раздела» for details on creating and enabling swap space.

Install the binaries, documentation, and shared libraries:

```
make install
```

Install the static libraries and headers:

```
make install-libs
```

Make the installed static libraries writable so debugging symbols can be removed later:

```
chmod -v u+w /usr/lib/{libcom_err,libe2p,libext2fs,libss}.a
```

This package installs a gzipped `.info` file but doesn't update the system-wide `dir` file. Unzip this file and then update the system `dir` file using the following commands.

```
gunzip -v /usr/share/info/libext2fs.info.gz
install-info --dir-file=/usr/share/info/dir \
             /usr/share/info/libext2fs.info
```

If desired, create and install some additional documentation by issuing the following commands:

```
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
```

6.21.2. Contents of E2fsprogs

Installed programs:	badblocks, chattr, compile_et, debugfs, dumpe2fs, e2freefrag, e2fsck, e2image, e2initrd_helper, e2label, e2undo, filefrag, fsck.ext2, fsck.ext3, fsck.ext4, fsck.ext4dev, logsave, lsattr, mk_cmds, mke2fs, mkfs.ext2, mkfs.ext3, mkfs.ext4, mkfs.ext4dev, mklost+found, resize2fs, and tune2fs
Installed libraries:	libcom_err.{a,so}, libe2p.{a,so}, libext2fs.{a,so} and libss.{a,so}
Installed directory:	/usr/include/e2p, /usr/include/et, /usr/include/ext2fs, /usr/include/ss, /usr/share/et, /usr/share/ss

Short Descriptions

badblocks	Searches a device (usually a disk partition) for bad blocks
chattr	Changes the attributes of files on an `ext2` file system; it also changes `ext3` file systems, the journaling version of `ext2` file systems
compile_et	An error table compiler; it converts a table of error-code names and messages into a C source file suitable for use with the `com_err` library
debugfs	A file system debugger; it can be used to examine and change the state of an `ext2` file system
dumpe2fs	Prints the super block and blocks group information for the file system present on a given device
e2freefrag	Reports free space fragmentation information
e2fsck	Is used to check, and optionally repair `ext2` file systems and `ext3` file systems
e2image	Is used to save critical `ext2` file system data to a file
e2initrd_helper	Prints the FS type of a given filesystem, given either a device name or label

e2label	Displays or changes the file system label on the ext2 file system present on a given device
e2undo	Replays the undo log undo_log for an ext2/ext3/ext4 filesystem found on a device. This can be used to undo a failed operation by an e2fsprogs program.
filefrag	Reports on how badly fragmented a particular file might be
fsck.ext2	By default checks ext2 file systems. This is a hard link to **e2fsck**.
fsck.ext3	By default checks ext3 file systems. This is a hard link to **e2fsck**.
fsck.ext4	By default checks ext4 file systems. This is a hard link to **e2fsck**.
fsck.ext4dev	By default checks ext4 development file systems. This is a hard link to **e2fsck**.
logsave	Saves the output of a command in a log file
lsattr	Lists the attributes of files on a second extended file system
mk_cmds	Converts a table of command names and help messages into a C source file suitable for use with the libss subsystem library
mke2fs	Creates an ext2 or ext3 file system on the given device
mkfs.ext2	By default creates ext2 file systems. This is a hard link to **mke2fs**.
mkfs.ext3	By default creates ext3 file systems. This is a hard link to **mke2fs**.
mkfs.ext4	By default creates ext4 file systems. This is a hard link to **mke2fs**.
mkfs.ext4dev	By default creates ext4 development file systems. This is a hard link to **mke2fs**.
mklost+found	Used to create a lost+found directory on an ext2 file system; it pre-allocates disk blocks to this directory to lighten the task of **e2fsck**
resize2fs	Can be used to enlarge or shrink an ext2 file system
tune2fs	Adjusts tunable file system parameters on an ext2 file system
libcom_err	The common error display routine
libe2p	Used by **dumpe2fs**, **chattr**, and **lsattr**
libext2fs	Contains routines to enable user-level programs to manipulate an ext2 file system
libss	Used by **debugfs**

6.22. Coreutils-8.10

The Coreutils package contains utilities for showing and setting the basic system characteristics.

Приблизительное время сборки:	3.2 SBU
Требует свободного места на диске:	99 MB

6.22.1. Installation of Coreutils

A known issue with the **uname** program from this package is that the *-p* switch always returns `unknown`. The following patch fixes this behavior for Intel architectures:

```
case `uname -m` in
 i?86 | x86_64) patch -Np1 -i ../coreutils-8.10-uname-1.patch ;;
esac
```

POSIX requires that programs from Coreutils recognize character boundaries correctly even in multibyte locales. The following patch fixes this non-compliance and other internationalization-related bugs:

```
patch -Np1 -i ../coreutils-8.10-i18n-1.patch
```

Замечание

In the past, many bugs were found in this patch. When reporting new bugs to Coreutils maintainers, please check first if they are reproducible without this patch.

Now prepare Coreutils for compilation:

```
./configure --prefix=/usr \
    --enable-no-install-program=kill,uptime
```

The meaning of the configure options:

`--enable-no-install-program=kill,uptime`
The purpose of this switch is to prevent Coreutils from installing binaries that will be installed by other packages later.

Compile the package:

```
make
```

Skip down to «Install the package» if not running the test suite.

Now the test suite is ready to be run. First, run the tests that are meant to be run as user `root`:

```
make NON_ROOT_USERNAME=nobody check-root
```

We're going to run the remainder of the tests as the `nobody` user. Certain tests, however, require that the user be a member of more than one group. So that these tests are not skipped we'll add a temporary group and make the user `nobody` a part of it:

```
echo "dummy:x:1000:nobody" >> /etc/group
```

Fix some of the permissions so that the non-root user can compile and run the tests:

```
chown -Rv nobody .
```

Now run the tests:

```
su-tools nobody -s /bin/bash -c "make RUN_EXPENSIVE_TESTS=yes check"
```

Remove the temporary group:

```
sed -i '/dummy/d' /etc/group
```

Install the package:

```
make install
```

Move programs to the locations specified by the 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
```

Some of the scripts in the LFS-Bootscripts package depend on **head**, **sleep**, and **nice**. As `/usr` may not be available during the early stages of booting, those binaries need to be on the root partition:

```
mv -v /usr/bin/{head,sleep,nice} /bin
```

6.22.2. Contents of Coreutils

Installed programs:	base64, basename, cat, chcon, chgrp, chmod, chown, chroot, cksum, comm, cp, csplit, cut, date, dd, df, dir, dircolors, dirname, du, echo, env, expand, expr, factor, false, fmt, fold, groups, head, hostid, id, install, join, link, ln, logname, ls, md5sum, mkdir, mkfifo, mknod, mktemp, mv, nice, nl, nohup, nproc, od, paste, pathchk, pinky, pr, printenv, printf, ptx, pwd, readlink, 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
Installed library:	libstdbuf.so
Installed directory:	/usr/lib/coreutils

Short Descriptions

base64	Encodes and decodes data according to the base64 (RFC 3548) specification
basename	Strips any path and a given suffix from a file name
cat	Concatenates files to standard output
chcon	Changes security context for files and directories
chgrp	Changes the group ownership of files and directories

chmod Changes the permissions of each file to the given mode; the mode can be either a symbolic representation of the changes to make or an octal number representing the new permissions

chown Changes the user and/or group ownership of files and directories

chroot Runs a command with the specified directory as the / directory

cksum Prints the Cyclic Redundancy Check (CRC) checksum and the byte counts of each specified file

comm Compares two sorted files, outputting in three columns the lines that are unique and the lines that are common

cp Copies files

csplit Splits a given file into several new files, separating them according to given patterns or line numbers and outputting the byte count of each new file

cut Prints sections of lines, selecting the parts according to given fields or positions

date Displays the current time in the given format, or sets the system date

dd Copies a file using the given block size and count, while optionally performing conversions on it

df Reports the amount of disk space available (and used) on all mounted file systems, or only on the file systems holding the selected files

dir Lists the contents of each given directory (the same as the **ls** command)

dircolors Outputs commands to set the LS_COLOR environment variable to change the color scheme used by **ls**

dirname Strips the non-directory suffix from a file name

du Reports the amount of disk space used by the current directory, by each of the given directories (including all subdirectories) or by each of the given files

echo Displays the given strings

env Runs a command in a modified environment

expand Converts tabs to spaces

expr Evaluates expressions

factor Prints the prime factors of all specified integer numbers

false Does nothing, unsuccessfully; it always exits with a status code indicating failure

fmt Reformats the paragraphs in the given files

fold Wraps the lines in the given files

groups Reports a user's group memberships

head Prints the first ten lines (or the given number of lines) of each given file

hostid Reports the numeric identifier (in hexadecimal) of the host

id Reports the effective user ID, group ID, and group memberships of the current user or specified user

install Copies files while setting their permission modes and, if possible, their owner and group

join Joins the lines that have identical join fields from two separate files

link Creates a hard link with the given name to a file

ln Makes hard links or soft (symbolic) links between files

logname Reports the current user's login name

ls Lists the contents of each given directory

md5sum Reports or checks Message Digest 5 (MD5) checksums

mkdir Creates directories with the given names

mkfifo Creates First-In, First-Outs (FIFOs), a «named pipe» in UNIX parlance, with the given names

mknod Creates device nodes with the given names; a device node is a character special file, a block special file, or a FIFO

mktemp Creates temporary files in a secure manner; it is used in scripts

mv Moves or renames files or directories

nice Runs a program with modified scheduling priority

nl Numbers the lines from the given files

nohup Runs a command immune to hangups, with its output redirected to a log file

nproc Prints the number of processing units available to a process

od Dumps files in octal and other formats

paste Merges the given files, joining sequentially corresponding lines side by side, separated by tab characters

pathchk Checks if file names are valid or portable

pinky Is a lightweight finger client; it reports some information about the given users

pr Paginates and columnates files for printing

printenv Prints the environment

printf Prints the given arguments according to the given format, much like the C printf function

ptx Produces a permuted index from the contents of the given files, with each keyword in its context

pwd Reports the name of the current working directory

readlink Reports the value of the given symbolic link

rm Removes files or directories

rmdir Removes directories if they are empty

runcon Runs a command with specified security context

seq Prints a sequence of numbers within a given range and with a given increment

sha1sum Prints or checks 160-bit Secure Hash Algorithm 1 (SHA1) checksums

sha224sum Prints or checks 224-bit Secure Hash Algorithm checksums

sha256sum Prints or checks 256-bit Secure Hash Algorithm checksums

sha384sum Prints or checks 384-bit Secure Hash Algorithm checksums

sha512sum Prints or checks 512-bit Secure Hash Algorithm checksums

shred Overwrites the given files repeatedly with complex patterns, making it difficult to recover the data

shuf Shuffles lines of text

sleep Pauses for the given amount of time

sort Sorts the lines from the given files

split Splits the given file into pieces, by size or by number of lines

stat Displays file or filesystem status

stdbuf Runs commands with altered buffering operations for its standard streams

stty Sets or reports terminal line settings

sum Prints checksum and block counts for each given file

sync Flushes file system buffers; it forces changed blocks to disk and updates the super block

tac Concatenates the given files in reverse

tail Prints the last ten lines (or the given number of lines) of each given file

tee Reads from standard input while writing both to standard output and to the given files

test Compares values and checks file types

timeout Runs a command with a time limit

touch Changes file timestamps, setting the access and modification times of the given files to the current time; files that do not exist are created with zero length

tr Translates, squeezes, and deletes the given characters from standard input

true Does nothing, successfully; it always exits with a status code indicating success

truncate Shrinks or expands a file to the specified size

tsort Performs a topological sort; it writes a completely ordered list according to the partial ordering in a given file

tty Reports the file name of the terminal connected to standard input

uname Reports system information

unexpand Converts spaces to tabs

uniq Discards all but one of successive identical lines

unlink Removes the given file

users Reports the names of the users currently logged on

vdir Is the same as **ls -l**

wc Reports the number of lines, words, and bytes for each given file, as well as a total line when more than one file is given

who Reports who is logged on

whoami Reports the user name associated with the current effective user ID

yes Repeatedly outputs «y» or a given string until killed

`libstdbuf` Library used by **stdbuf**

6.23. Iana-Etc-2.30

The Iana-Etc package provides data for network services and protocols.

Приблизительное время сборки:	less than 0.1 SBU
Требует свободного места на диске:	2.3 MB

6.23.1. Installation of Iana-Etc

The following command converts the raw data provided by IANA into the correct formats for the `/etc/protocols` and `/etc/services` data files:

```
make
```

This package does not come with a test suite.

Install the package:

```
make install
```

6.23.2. Contents of Iana-Etc

Installed files: /etc/protocols and /etc/services

Short Descriptions

`/etc/protocols`	Describes the various DARPA Internet protocols that are available from the TCP/IP subsystem
`/etc/services`	Provides a mapping between friendly textual names for internet services, and their underlying assigned port numbers and protocol types

6.24. M4-1.4.15

The M4 package contains a macro processor.

Приблизительное время сборки:	0.4 SBU
Требует свободного места на диске:	14.2 MB

6.24.1. Installation of M4

Prepare M4 for compilation:

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

Compile the package:

```
make
```

To test the results, issue:

```
make check
```

Install the package:

```
make install
```

6.24.2. Contents of M4

Installed program: m4

Short Descriptions

m4 copies the given files while expanding the macros that they contain. These macros are either built-in or user-defined and can take any number of arguments. Besides performing macro expansion, **m4** has built-in functions for including named files, running Unix commands, performing integer arithmetic, manipulating text, recursion, etc. The **m4** program can be used either as a front-end to a compiler or as a macro processor in its own right.

6.25. Bison-2.4.3

The Bison package contains a parser generator.

Приблизительное время сборки:	1.1 SBU
Требует свободного места на диске:	19.2 MB

6.25.1. Installation of Bison

Prepare Bison for compilation:

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

The configure system causes Bison to be built without support for internationalization of error messages if a **bison** program is not already in $PATH. The following addition will correct this:

```
echo '#define YYENABLE_NLS 1' >> lib/config.h
```

Compile the package:

```
make
```

To test the results (about 0.5 SBU), issue:

```
make check
```

Install the package:

```
make install
```

6.25.2. Contents of Bison

Installed programs:	bison and yacc
Installed library:	liby.a
Installed directory:	/usr/share/bison

Short Descriptions

bison Generates, from a series of rules, a program for analyzing the structure of text files; Bison is a replacement for Yacc (Yet Another Compiler Compiler)

yacc A wrapper for **bison**, meant for programs that still call **yacc** instead of **bison**; it calls **bison** with the *-y* option

`liby.a` The Yacc library containing implementations of Yacc-compatible `yyerror` and `main` functions; this library is normally not very useful, but POSIX requires it

6.26. Procps-3.2.8

The Procps package contains programs for monitoring processes.

Приблизительное время сборки:	0.1 SBU
Требует свободного места на диске:	2.3 MB

6.26.1. Installation of Procps

Apply a patch to prevent an error message from being displayed when determining the kernel clock tick rate:

```
patch -Np1 -i ../procps-3.2.8-fix_HZ_errors-1.patch
```

Apply a patch to fix a unicode related issue in the **watch** program:

```
patch -Np1 -i ../procps-3.2.8-watch_unicode-1.patch
```

Fix a bug in the Makefile, which prevents procps from building with make-3.82:

```
sed -i -e 's@\*/module.mk@proc/module.mk ps/module.mk@' Makefile
```

Compile the package:

```
make
```

This package does not come with a test suite.

Install the package:

```
make install
```

6.26.2. Contents of Procps

Installed programs:	free, kill, pgrep, pkill, pmap, ps, pwdx, skill, slabtop, snice, sysctl, tload, top, uptime, vmstat, w, and watch
Installed library:	libproc.so

Short Descriptions

free	Reports the amount of free and used memory (both physical and swap memory) in the system
kill	Sends signals to processes
pgrep	Looks up processes based on their name and other attributes
pkill	Signals processes based on their name and other attributes
pmap	Reports the memory map of the given process
ps	Lists the current running processes
pwdx	Reports the current working directory of a process
skill	Sends signals to processes matching the given criteria
slabtop	Displays detailed kernel slap cache information in real time

snice	Changes the scheduling priority of processes matching the given criteria
sysctl	Modifies kernel parameters at run time
tload	Prints a graph of the current system load average
top	Displays a list of the most CPU intensive processes; it provides an ongoing look at processor activity in real time
uptime	Reports how long the system has been running, how many users are logged on, and the system load averages
vmstat	Reports virtual memory statistics, giving information about processes, memory, paging, block Input/Output (IO), traps, and CPU activity
w	Shows which users are currently logged on, where, and since when
watch	Runs a given command repeatedly, displaying the first screen-full of its output; this allows a user to watch the output change over time
`libproc`	Contains the functions used by most programs in this package

6.27. Grep-2.7

The Grep package contains programs for searching through files.

Приблизительное время сборки:	0.1 SBU
Требует свободного места на диске:	7.3 MB

6.27.1. Installation of Grep

Prepare Grep for compilation:

```
./configure --prefix=/usr \
    --bindir=/bin
```

Compile the package:

```
make
```

To test the results, issue:

```
make check
```

Install the package:

```
make install
```

6.27.2. Contents of Grep

Installed programs:	egrep, fgrep, and grep

Short Descriptions

egrep Prints lines matching an extended regular expression

fgrep Prints lines matching a list of fixed strings

grep Prints lines matching a basic regular expression

6.28. Readline-6.2

The Readline package is a set of libraries that offers command-line editing and history capabilities.

Приблизительное время сборки:	0.2 SBU
Требует свободного места на диске:	13.8 MB

6.28.1. Installation of Readline

Reinstalling Readline will cause the old libraries to be moved to <libraryname>.old. While this is normally not a problem, in some cases it can trigger a linking bug in **ldconfig**. This can be avoided by issuing the following two seds:

```
sed -i '/MV.*old/d' Makefile.in
sed -i '/{OLDSUFF}/c:' support/shlib-install
```

Prepare Readline for compilation:

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

Compile the package:

```
make SHLIB_LIBS=-lncurses
```

The meaning of the make option:

SHLIB_LIBS=-lncurses
: This option forces Readline to link against the `libncurses` (really, `libncursesw`) library.

This package does not come with a test suite.

Install the package:

```
make install
```

Now move the static libraries to a more appropriate location:

```
mv -v /lib/lib{readline,history}.a /usr/lib
```

Next, remove the `.so` files in `/lib` and relink them into `/usr/lib`:

```
rm -v /lib/lib{readline,history}.so
ln -sfv ../../lib/libreadline.so.6 /usr/lib/libreadline.so
ln -sfv ../../lib/libhistory.so.6 /usr/lib/libhistory.so
```

If desired, install the documentation:

```
mkdir   -v       /usr/share/doc/readline-6.2
install -v -m644 doc/*.{ps,pdf,html,dvi} \
                 /usr/share/doc/readline-6.2
```

6.28.2. Contents of Readline

Installed libraries: libhistory.{a,so}, and libreadline.{a,so}

Installed directories: /usr/include/readline, /usr/share/readline, /usr/share/doc/readline-6.2

Short Descriptions

`libhistory` Provides a consistent user interface for recalling lines of history

`libreadline` Aids in the consistency of user interface across discrete programs that need to provide a command line interface

6.29. Bash-4.2

The Bash package contains the Bourne-Again SHell.

Приблизительное время сборки:	1.4 SBU
Требует свободного места на диске:	35 MB

6.29.1. Installation of Bash

Prepare Bash for compilation:

```
./configure --prefix=/usr --bindir=/bin \
    --htmldir=/usr/share/doc/bash-4.2 --without-bash-malloc \
    --with-installed-readline
```

The meaning of the configure options:

`--htmldir`
This option designates the directory into which HTML formatted documentation will be installed.

`--with-installed-readline`
This option tells Bash to use the `readline` library that is already installed on the system rather than using its own readline version.

Compile the package:

```
make
```

Skip down to «Install the package» if not running the test suite.

To prepare the tests, ensure that the `nobody` user can write to the sources tree:

```
chown -Rv nobody .
```

Now, run the tests as the `nobody` user:

```
su-tools nobody -s /bin/bash -c "make tests"
```

Install the package:

```
make install
```

Run the newly compiled **bash** program (replacing the one that is currently being executed):

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

Замечание

The parameters used make the **bash** process an interactive login shell and continue to disable hashing so that new programs are found as they become available.

6.29.2. Contents of Bash

Installed programs: bash, bashbug, and sh (link to bash)
Installed directory: /usr/share/doc/bash-4.2

Short Descriptions

bash A widely-used command interpreter; it performs many types of expansions and substitutions on a given command line before executing it, thus making this interpreter a powerful tool

bashbug A shell script to help the user compose and mail standard formatted bug reports concerning **bash**

sh A symlink to the **bash** program; when invoked as **sh**, **bash** tries to mimic the startup behavior of historical versions of **sh** as closely as possible, while conforming to the POSIX standard as well

6.30. Libtool-2.4

The Libtool package contains the GNU generic library support script. It wraps the complexity of using shared libraries in a consistent, portable interface.

Приблизительное время сборки:	3.7 SBU
Требует свободного места на диске:	35 MB

6.30.1. Installation of Libtool

Prepare Libtool for compilation:

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

Compile the package:

```
make
```

To test the results (about 3.0 SBU), issue:

```
make check
```

Install the package:

```
make install
```

6.30.2. Contents of Libtool

Installed programs:	libtool and libtoolize
Installed libraries:	libltdl.{a,so}
Installed directories:	/usr/include/libltdl, /usr/share/libtool

Short Descriptions

libtool	Provides generalized library-building support services
libtoolize	Provides a standard way to add **libtool** support to a package
`libltdl`	Hides the various difficulties of dlopening libraries

6.31. GDBM-1.8.3

The GDBM package contains the GNU Database Manager. This is a disk file format database which stores key/data-pairs in single files. The actual data of any record being stored is indexed by a unique key, which can be retrieved in less time than if it was stored in a text file.

Приблизительное время сборки:	0.1 SBU
Требует свободного места на диске:	2.7 MB

6.31.1. Installation of GDBM

Prepare GDBM for compilation:

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

Compile the package:

```
make
```

This package does not come with a test suite.

Install the package:

```
make install
```

In addition, install the DBM and NDBM compatibility headers, as some packages outside of LFS may look for these older dbm routines:

```
make install-compat
```

Fix a minor installation issue by manually adding GDBM to the **info** table of contents:

```
install-info --dir-file=/usr/info/dir /usr/info/gdbm.info
```

6.31.2. Contents of GDBM

Installed libraries: libgdbm.{so,a} and libgdbm_compat.{so,a}

Short Descriptions

`libgdbm` Contains functions to manipulate a hashed database

6.32. Inetutils-1.8

The Inetutils package contains programs for basic networking.

Приблизительное время сборки:	0.4 SBU
Требует свободного места на диске:	17 MB

6.32.1. Installation of Inetutils

```
./configure --prefix=/usr --libexecdir=/usr/sbin \
    --localstatedir=/var --disable-ifconfig \
    --disable-logger --disable-syslogd --disable-whois \
    --disable-servers
```

The meaning of the configure options:

`--disable-ifconfig`
This option prevents Inetutils from installing the **ifconfig** program, which can be used to configure network interfaces. LFS uses **ip** from IPRoute2 to perform this task.

`--disable-logger`
This option prevents Inetutils from installing the **logger** program, which is used by scripts to pass messages to the System Log Daemon. Do not install it because Util-linux installed a version earlier.

`--disable-syslogd`
This option prevents Inetutils from installing the System Log Daemon, which is installed with the Sysklogd package.

`--disable-whois`
This option disables the building of the Inetutils **whois** client, which is out of date. Instructions for a better **whois** client are in the BLFS book.

`--disable-servers`
This disables the installation of the various network servers included as part of the Inetutils package. These servers are deemed not appropriate in a basic LFS system. Some are insecure by nature and are only considered safe on trusted networks. More information can be found at *http://www.linuxfromscratch.org/blfs/view/svn/basicnet/inetutils.html*. Note that better replacements are available for many of these servers.

Compile the package:

```
make
```

To test the results, issue:

```
make check
```

Install the package:

```
make install
make -C doc html
make -C doc install-html docdir=/usr/share/doc/inetutils-1.8
```

Move some programs to their FHS-compliant place:

```
mv -v /usr/bin/{hostname,ping,ping6} /bin
mv -v /usr/bin/traceroute /sbin
```

6.32.2. Contents of Inetutils

Installed programs: ftp, hostname, ping, ping6, rcp, rexec, rlogin, rsh, talk, telnet, tftp, and traceroute

Short Descriptions

ftp Is the file transfer protocol program

hostname Reports or sets the name of the host

ping Sends echo-request packets and reports how long the replies take

ping6 A version of **ping** for IPv6 networks

rcp Performs remote file copy

rexec executes commands on a remote host

rlogin Performs remote login

rsh Runs a remote shell

talk Is used to chat with another user

telnet An interface to the TELNET protocol

tftp A trivial file transfer program

traceroute Traces the route your packets take from the host you are working on to another host on a network, showing all the intermediate hops (gateways) along the way

6.33. Perl-5.12.3

The Perl package contains the Practical Extraction and Report Language.

Приблизительное время сборки:	5.5 SBU
Требует свободного места на диске:	171 MB

6.33.1. Installation of Perl

First create a basic `/etc/hosts` file to be referenced in one of Perl's configuration files as well as the optional test suite:

```
echo "127.0.0.1 localhost $(hostname)" > /etc/hosts
```

This version of Perl now builds the Compress::Raw::Zlib module. By default Perl will use an internal copy of the Zlib source for the build. Issue the following command so that Perl will use the Zlib library installed on the system:

```
sed -i -e "s|BUILD_ZLIB\s*= True|BUILD_ZLIB = False|"           \
       -e "s|INCLUDE\s*= ./zlib-src|INCLUDE    = /usr/include|" \
       -e "s|LIB\s*= ./zlib-src|LIB        = /usr/lib|"        \
    cpan/Compress-Raw-Zlib/config.in
```

To have full control over the way Perl is set up, you can remove the «-des» options from the following command and hand-pick the way this package is built. Alternatively, use the command exactly as below to use the defaults that Perl auto-detects:

```
sh Configure -des -Dprefix=/usr \
                  -Dvendorprefix=/usr           \
                  -Dman1dir=/usr/share/man/man1 \
                  -Dman3dir=/usr/share/man/man3 \
                  -Dpager="/usr/bin/less -isR"  \
                  -Duseshrplib
```

The meaning of the configure options:

`-Dvendorprefix=/usr`
This ensures **perl** knows how to tell packages where they should install their perl modules.

`-Dpager="/usr/bin/less -isR"`
This corrects an error in the way that **perldoc** invokes the **less** program.

`-Dman1dir=/usr/share/man/man1 -Dman3dir=/usr/share/man/man3`
Since Groff is not installed yet, **Configure** thinks that we do not want man pages for Perl. Issuing these parameters overrides this decision.

`-Duseshrplib`
Build a shared libperl needed by some perl modules.

Compile the package:

```
make
```

To test the results (approximately 2.5 SBU), issue:

```
make test
```

Install the package:

```
make install
```

6.33.2. Contents of Perl

Installed programs:	a2p, c2ph, config_data, corelist, cpan, cpan2dist, cpanp, cpanp-run-perl, dprofpp, enc2xs, find2perl, h2ph, h2xs, instmodsh, libnetcfg, perl, perl5.12.3 (link to perl), perlbug, perldoc, perlivp, perlthanks (link to perlbug), piconv, pl2pm, pod2html, pod2latex, pod2man, pod2text, pod2usage, podchecker, podselect, prove, psed (link to s2p), pstruct (link to c2ph), ptar, ptardiff, s2p, shasum, splain, and xsubpp
Installed libraries:	Several hundred which cannot all be listed here
Installed directory:	/usr/lib/perl5

Short Descriptions

a2p	Translates awk to Perl
c2ph	Dumps C structures as generated from **cc -g -S**
config_data	Queries or changes configuration of Perl modules
corelist	A commandline frontend to Module::CoreList
cpan	Interact with the Comprehensive Perl Archive Network (CPAN) from the command line
cpan2dist	The CPANPLUS distribution creator
cpanp	The CPANPLUS launcher
cpanp-run-perl	Perl script that is used to enable flushing of the output buffer after each write in spawned processes
dprofpp	Displays Perl profile data
enc2xs	Builds a Perl extension for the Encode module from either Unicode Character Mappings or Tcl Encoding Files
find2perl	Translates **find** commands to Perl
h2ph	Converts `.h` C header files to `.ph` Perl header files
h2xs	Converts `.h` C header files to Perl extensions
instmodsh	Shell script for examining installed Perl modules, and can even create a tarball from an installed module
libnetcfg	Can be used to configure the `libnet` Perl module
perl	Combines some of the best features of C, **sed**, **awk** and **sh** into a single swiss-army language
perl5.12.3	A hard link to **perl**
perlbug	Used to generate bug reports about Perl, or the modules that come with it, and mail them
perldoc	Displays a piece of documentation in pod format that is embedded in the Perl installation tree or in a Perl script

perlivp	The Perl Installation Verification Procedure; it can be used to verify that Perl and its libraries have been installed correctly
perlthanks	Used to generate thank you messages to mail to the Perl developers
piconv	A Perl version of the character encoding converter **iconv**
pl2pm	A rough tool for converting Perl4 `.pl` files to Perl5 `.pm` modules
pod2html	Converts files from pod format to HTML format
pod2latex	Converts files from pod format to LaTeX format
pod2man	Converts pod data to formatted *roff input
pod2text	Converts pod data to formatted ASCII text
pod2usage	Prints usage messages from embedded pod docs in files
podchecker	Checks the syntax of pod format documentation files
podselect	Displays selected sections of pod documentation
prove	Command line tool for running tests against the Test::Harness module.
psed	A Perl version of the stream editor **sed**
pstruct	Dumps C structures as generated from **cc -g -S** stabs
ptar	A **tar**-like program written in Perl
ptardiff	A Perl program that compares an extracted archive with an unextracted one
s2p	Translates **sed** scripts to Perl
shasum	Prints or checks SHA checksums
splain	Is used to force verbose warning diagnostics in Perl
xsubpp	Converts Perl XS code into C code

6.34. Autoconf-2.68

The Autoconf package contains programs for producing shell scripts that can automatically configure source code.

Приблизительное время сборки:	4.8 SBU
Требует свободного места на диске:	12.4 MB

6.34.1. Installation of Autoconf

Prepare Autoconf for compilation:

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

Compile the package:

```
make
```

To test the results, issue:

```
make check
```

This takes a long time, about 4.7 SBUs. In addition, 6 tests are skipped that use Automake. For full test coverage, Autoconf can be re-tested after Automake has been installed.

Install the package:

```
make install
```

6.34.2. Contents of Autoconf

Installed programs:	autoconf, autoheader, autom4te, autoreconf, autoscan, autoupdate, and ifnames
Installed directory:	/usr/share/autoconf

Short Descriptions

autoconf	Produces shell scripts that automatically configure software source code packages to adapt to many kinds of Unix-like systems. The configuration scripts it produces are independent—running them does not require the **autoconf** program.
autoheader	A tool for creating template files of C *#define* statements for configure to use
autom4te	A wrapper for the M4 macro processor
autoreconf	Automatically runs **autoconf**, **autoheader**, **aclocal**, **automake**, **gettextize**, and **libtoolize** in the correct order to save time when changes are made to **autoconf** and **automake** template files
autoscan	Helps to create a `configure.in` file for a software package; it examines the source files in a directory tree, searching them for common portability issues, and creates a `configure.scan` file that serves as as a preliminary `configure.in` file for the package

autoupdate Modifies a `configure.in` file that still calls **autoconf** macros by their old names to use the current macro names

ifnames Helps when writing `configure.in` files for a software package; it prints the identifiers that the package uses in C preprocessor conditionals. If a package has already been set up to have some portability, this program can help determine what **configure** needs to check for. It can also fill in gaps in a `configure.in` file generated by **autoscan**

6.35. Automake-1.11.1

The Automake package contains programs for generating Makefiles for use with Autoconf.

Приблизительное время сборки:	18.3 SBU
Требует свободного места на диске:	28.8 MB

6.35.1. Installation of Automake

Prepare Automake for compilation:

```
./configure --prefix=/usr --docdir=/usr/share/doc/automake-1.11.1
```

Compile the package:

```
make
```

To test the results, issue:

```
make check
```

This takes a long time, about 10 SBUs.

Install the package:

```
make install
```

6.35.2. Contents of Automake

Installed programs:	acinstall, aclocal, aclocal-1.11.1, automake, automake-1.11.1, compile, config.guess, config.sub, depcomp, elisp-comp, install-sh, mdate-sh, missing, mkinstalldirs, py-compile, symlink-tree, and ylwrap
Installed directories:	/usr/share/aclocal-1.11, /usr/share/automake-1.11, /usr/share/doc/automake-1.11.1

Short Descriptions

acinstall	A script that installs aclocal-style M4 files
aclocal	Generates `aclocal.m4` files based on the contents of `configure.in` files
aclocal-1.11.1	A hard link to **aclocal**
automake	A tool for automatically generating `Makefile.in` files from `Makefile.am` files. To create all the `Makefile.in` files for a package, run this program in the top-level directory. By scanning the `configure.in` file, it automatically finds each appropriate `Makefile.am` file and generates the corresponding `Makefile.in` file
automake-1.11.1	A hard link to **automake**
compile	A wrapper for compilers
config.guess	A script that attempts to guess the canonical triplet for the given build, host, or target architecture

config.sub	A configuration validation subroutine script
depcomp	A script for compiling a program so that dependency information is generated in addition to the desired output
elisp-comp	Byte-compiles Emacs Lisp code
install-sh	A script that installs a program, script, or data file
mdate-sh	A script that prints the modification time of a file or directory
missing	A script acting as a common stub for missing GNU programs during an installation
mkinstalldirs	A script that creates a directory tree
py-compile	Compiles a Python program
symlink-tree	A script to create a symlink tree of a directory tree
ylwrap	A wrapper for **lex** and **yacc**

6.36. Bzip2-1.0.6

The Bzip2 package contains programs for compressing and decompressing files. Compressing text files with **bzip2** yields a much better compression percentage than with the traditional **gzip**.

Приблизительное время сборки:	less than 0.1 SBU
Требует свободного места на диске:	6.4 MB

6.36.1. Installation of Bzip2

Apply a patch that will install the documentation for this package:

```
patch -Np1 -i ../bzip2-1.0.6-install_docs-1.patch
```

The following command ensures installation of symbolic links are relative:

```
sed -i 's@\(ln -s -f \)$(PREFIX)/bin/@\1@' Makefile
```

Prepare Bzip2 for compilation with:

```
make -f Makefile-libbz2_so
make clean
```

The meaning of the make parameter:

`-f Makefile-libbz2_so`
This will cause Bzip2 to be built using a different `Makefile` file, in this case the `Makefile-libbz2_so` file, which creates a dynamic `libbz2.so` library and links the Bzip2 utilities against it.

Compile and test the package:

```
make
```

Install the programs:

```
make PREFIX=/usr install
```

Install the shared **bzip2** binary into the `/bin` directory, make some necessary symbolic links, and clean up:

```
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
```

6.36.2. Contents of Bzip2

Installed programs:	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
Installed libraries:	libbz2.{a,so}
Installed directory:	/usr/share/doc/bzip2-1.0.6

Short Descriptions

bunzip2	Decompresses bzipped files
bzcat	Decompresses to standard output
bzcmp	Runs **cmp** on bzipped files
bzdiff	Runs **diff** on bzipped files
bzegrep	Runs **egrep** on bzipped files
bzfgrep	Runs **fgrep** on bzipped files
bzgrep	Runs **grep** on bzipped files
bzip2	Compresses files using the Burrows-Wheeler block sorting text compression algorithm with Huffman coding; the compression rate is better than that achieved by more conventional compressors using «Lempel-Ziv» algorithms, like **gzip**
bzip2recover	Tries to recover data from damaged bzipped files
bzless	Runs **less** on bzipped files
bzmore	Runs **more** on bzipped files
`libbz2*`	The library implementing lossless, block-sorting data compression, using the Burrows-Wheeler algorithm

6.37. Diffutils-3.0

The Diffutils package contains programs that show the differences between files or directories.

Приблизительное время сборки:	0.1 SBU
Требует свободного места на диске:	6.3 MB

6.37.1. Installation of Diffutils

Prepare Diffutils for compilation:

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

Compile the package:

```
make
```

To test the results, issue:

```
make check
```

Install the package:

```
make install
```

6.37.2. Contents of Diffutils

Installed programs:	cmp, diff, diff3, and sdiff

Short Descriptions

cmp Compares two files and reports whether or in which bytes they differ

diff Compares two files or directories and reports which lines in the files differ

diff3 Compares three files line by line

sdiff Merges two files and interactively outputs the results

6.38. Gawk-3.1.8

The Gawk package contains programs for manipulating text files.

Приблизительное время сборки:	0.2 SBU
Требует свободного места на диске:	19 MB

6.38.1. Installation of Gawk

Prepare Gawk for compilation:

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

Compile the package:

```
make
```

To test the results, issue:

```
make check
```

Install the package:

```
make install
```

If desired, install the documentation:

```
mkdir -v /usr/share/doc/gawk-3.1.8
cp    -v doc/{awkforai.txt,*.{eps,pdf,jpg}} \
         /usr/share/doc/gawk-3.1.8
```

6.38.2. Contents of Gawk

Installed programs:	awk (link to gawk), gawk, gawk-3.1.8, grcat, igawk, pgawk, pgawk-3.1.8, and pwcat
Installed directories:	/usr/lib/awk, /usr/share/awk

Short Descriptions

awk	A link to **gawk**
gawk	A program for manipulating text files; it is the GNU implementation of **awk**
gawk-3.1.8	A hard link to **gawk**
grcat	Dumps the group database `/etc/group`
igawk	Gives **gawk** the ability to include files
pgawk	The profiling version of **gawk**
pgawk-3.1.8	Hard link to **pgawk**
pwcat	Dumps the password database `/etc/passwd`

6.39. File-5.05

The File package contains a utility for determining the type of a given file or files.

Приблизительное время сборки:	0.2 SBU
Требует свободного места на диске:	9.5 MB

6.39.1. Installation of File

Prepare File for compilation:

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

Compile the package:

```
make
```

To test the results, issue:

```
make check
```

Install the package:

```
make install
```

6.39.2. Contents of File

Installed programs:	file
Installed library:	libmagic.{a,so}

Short Descriptions

file	Tries to classify each given file; it does this by performing several tests—file system tests, magic number tests, and language tests
`libmagic`	Contains routines for magic number recognition, used by the **file** program

6.40. Findutils-4.4.2

The Findutils package contains programs to find files. These programs are provided to recursively search through a directory tree and to create, maintain, and search a database (often faster than the recursive find, but unreliable if the database has not been recently updated).

Приблизительное время сборки:	0.5 SBU
Требует свободного места на диске:	22 MB

6.40.1. Installation of Findutils

Prepare Findutils for compilation:

```
./configure --prefix=/usr --libexecdir=/usr/lib/findutils \
    --localstatedir=/var/lib/locate
```

The meaning of the configure options:

`--localstatedir`
This option changes the location of the **locate** database to be in `/var/lib/locate`, which is FHS-compliant.

Compile the package:

```
make
```

To test the results, issue:

```
make check
```

Install the package:

```
make install
```

Some of the scripts in the LFS-Bootscripts package depend on **find**. As `/usr` may not be available during the early stages of booting, this program needs to be on the root partition. The **updatedb** script also needs to be modified to correct an explicit path:

```
mv -v /usr/bin/find /bin
sed -i 's/find:=${BINDIR}/find:=\/bin/' /usr/bin/updatedb
```

6.40.2. Contents of Findutils

Installed programs:	bigram, code, find, frcode, locate, oldfind, updatedb, and xargs
Installed directory:	/usr/lib/findutils

Short Descriptions

bigram	Was formerly used to produce **locate** databases
code	Was formerly used to produce **locate** databases; it is the ancestor of **frcode**.
find	Searches given directory trees for files matching the specified criteria

frcode Is called by **updatedb** to compress the list of file names; it uses front-compression, reducing the database size by a factor of four to five.

locate Searches through a database of file names and reports the names that contain a given string or match a given pattern

oldfind Older version of find, using a different algorithm

updatedb Updates the **locate** database; it scans the entire file system (including other file systems that are currently mounted, unless told not to) and puts every file name it finds into the database

xargs Can be used to apply a given command to a list of files

6.41. Flex-2.5.35

The Flex package contains a utility for generating programs that recognize patterns in text.

Приблизительное время сборки:	0.7 SBU
Требует свободного места на диске:	28 MB

6.41.1. Installation of Flex

Apply a patch that fixes a bug in the C++ scanner generator, that causes scanner compilation to fail when using GCC-4.5.2:

```
patch -Np1 -i ../flex-2.5.35-gcc44-1.patch
```

Prepare Flex for compilation:

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

Compile the package:

```
make
```

To test the results (about 0.5 SBU), issue:

```
make check
```

Install the package:

```
make install
```

There are some packages that expect to find the `lex` library in `/usr/lib`. Create a symlink to account for this:

```
ln -sv libfl.a /usr/lib/libl.a
```

A few programs do not know about **flex** yet and try to run its predecessor, **lex**. To support those programs, create a wrapper script named `lex` that calls `flex` in **lex** emulation mode:

```
cat > /usr/bin/lex << "EOF"
#!/bin/sh
# Begin /usr/bin/lex

exec /usr/bin/flex -l "$@"

# End /usr/bin/lex
EOF
chmod -v 755 /usr/bin/lex
```

If desired, install the `flex.pdf` documentation file:

```
mkdir -v /usr/share/doc/flex-2.5.35
cp    -v doc/flex.pdf \
         /usr/share/doc/flex-2.5.35
```

6.41.2. Contents of Flex

Installed programs: flex and lex
Installed libraries: libfl.a and libfl_pic.a

Short Descriptions

flex A tool for generating programs that recognize patterns in text; it allows for the versatility to specify the rules for pattern-finding, eradicating the need to develop a specialized program

lex A script that runs **flex** in **lex** emulation mode

`libfl.a` The `flex` library

6.42. Gettext-0.18.1.1

The Gettext package contains utilities for internationalization and localization. These allow programs to be compiled with NLS (Native Language Support), enabling them to output messages in the user's native language.

Приблизительное время сборки:	5.8 SBU
Требует свободного места на диске:	125 MB

6.42.1. Installation of Gettext

Prepare Gettext for compilation:

```
./configure --prefix=/usr \
            --docdir=/usr/share/doc/gettext-0.18.1.1
```

Compile the package:

```
make
```

To test the results (this takes a long time, around 3 SBUs), issue:

```
make check
```

Install the package:

```
make install
```

6.42.2. Contents of Gettext

Installed programs:	autopoint, config.charset, config.rpath, envsubst, gettext, gettext.sh, gettextize, hostname, msgattrib, msgcat, msgcmp, msgcomm, msgconv, msgen, msgexec, msgfilter, msgfmt, msggrep, msginit, msgmerge, msgunfmt, msguniq, ngettext, recode-sr-latin, and xgettext
Installed libraries:	libasprintf.{a,so}, libgettextlib.so, libgettextpo.{a,so}, libgettextsrc.so, and preloadable_libintl.so
Installed directories:	/usr/lib/gettext, /usr/share/doc/gettext-0.18.1.1, /usr/share/gettext

Short Descriptions

autopoint	Copies standard Gettext infrastructure files into a source package
config.charset	Outputs a system-dependent table of character encoding aliases
config.rpath	Outputs a system-dependent set of variables, describing how to set the runtime search path of shared libraries in an executable
envsubst	Substitutes environment variables in shell format strings
gettext	Translates a natural language message into the user's language by looking up the translation in a message catalog
gettext.sh	Primarily serves as a shell function library for gettext

gettextize	Copies all standard Gettext files into the given top-level directory of a package to begin internationalizing it
hostname	Displays a network hostname in various forms
msgattrib	Filters the messages of a translation catalog according to their attributes and manipulates the attributes
msgcat	Concatenates and merges the given `.po` files
msgcmp	Compares two `.po` files to check that both contain the same set of msgid strings
msgcomm	Finds the messages that are common to to the given `.po` files
msgconv	Converts a translation catalog to a different character encoding
msgen	Creates an English translation catalog
msgexec	Applies a command to all translations of a translation catalog
msgfilter	Applies a filter to all translations of a translation catalog
msgfmt	Generates a binary message catalog from a translation catalog
msggrep	Extracts all messages of a translation catalog that match a given pattern or belong to some given source files
msginit	Creates a new `.po` file, initializing the meta information with values from the user's environment
msgmerge	Combines two raw translations into a single file
msgunfmt	Decompiles a binary message catalog into raw translation text
msguniq	Unifies duplicate translations in a translation catalog
ngettext	Displays native language translations of a textual message whose grammatical form depends on a number
recode-sr-latin	Recodes Serbian text from Cyrillic to Latin script
xgettext	Extracts the translatable message lines from the given source files to make the first translation template
`libasprintf`	defines the *autosprintf* class, which makes C formatted output routines usable in C++ programs, for use with the <*string*> strings and the <*iostream*> streams
`libgettextlib`	a private library containing common routines used by the various Gettext programs; these are not intended for general use
`libgettextpo`	Used to write specialized programs that process `.po` files; this library is used when the standard applications shipped with Gettext (such as **msgcomm**, **msgcmp**, **msgattrib**, and **msgen**) will not suffice
`libgettextsrc`	A private library containing common routines used by the various Gettext programs; these are not intended for general use
`preloadable_libintl`	A library, intended to be used by LD_PRELOAD that assists `libintl` in logging untranslated messages.

6.43. Groff-1.21

The Groff package contains programs for processing and formatting text.

Приблизительное время сборки:	0.4 SBU
Требует свободного места на диске:	78 MB

6.43.1. Installation of Groff

Groff expects the environment variable PAGE to contain the default paper size. For users in the United States, *PAGE=letter* is appropriate. Elsewhere, *PAGE=A4* may be more suitable. While the default paper size is configured during compilation, it can be overridden later by echoing either «A4» or «letter» to the /etc/papersize file.

Prepare Groff for compilation:

```
PAGE=<paper_size> ./configure --prefix=/usr
```

Compile the package:

```
make
```

This package does not come with a test suite.

Install the package:

```
make install
```

Some documentation programs, such as **xman**, will not work properly without the following symlinks:

```
ln -sv eqn /usr/bin/geqn
ln -sv tbl /usr/bin/gtbl
```

6.43.2. Contents of Groff

Installed programs:	addftinfo, afmtodit, chem, eqn, eqn2graph, gdiffmk, geqn (link to eqn), grap2graph, grn, grodvi, groff, groffer, grog, grolbp, grolj4, grops, grotty, gtbl (link to tbl), hpftodit, indxbib, lkbib, lookbib, mmroff, neqn, nroff, pdfroff, pfbtops, pic, pic2graph, post-grohtml, preconv, pre-grohtml, refer, roff2dvi, roff2html, roff2pdf, roff2ps, roff2text, roff2x, soelim, tbl, tfmtodit, and troff
Installed directories:	/usr/lib/groff, /usr/share/doc/groff-1.21, /usr/share/groff

Short Descriptions

addftinfo	Reads a troff font file and adds some additional font-metric information that is used by the **groff** system
afmtodit	Creates a font file for use with **groff** and **grops**
chem	Groff preprocessor for producing chemical structure diagrams
eqn	Compiles descriptions of equations embedded within troff input files into commands that are understood by **troff**

eqn2graph	Converts a troff EQN (equation) into a cropped image
gdiffmk	Marks differences between groff/nroff/troff files
geqn	A link to **eqn**
grap2graph	Converts a grap diagram into a cropped bitmap image
grn	A **groff** preprocessor for gremlin files
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
grops	Translates the output of GNU **troff** to PostScript
grotty	Translates the output of GNU **troff** into a form suitable for typewriter-like devices
gtbl	A link to **tbl**
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**
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

6.44. GRUB-1.98

The GRUB package contains the GRand Unified Bootloader.

Приблизительное время сборки:	0.4 SBU
Требует свободного места на диске:	27.6 MB

6.44.1. Installation of GRUB

Prepare GRUB for compilation:

```
./configure --prefix=/usr           \
            --sysconfdir=/etc       \
            --disable-grub-emu-usb  \
            --disable-grub-fstest   \
            --disable-efiemu
```

The --disable switches minimize what is built by disabling features and testing programs not really needed for LFS.

Compile the package:

```
make
```

This package does not come with a test suite.

Install the package:

```
make install
```

Using GRUB to make your LFS system bootable will be discussed in Раздел 8.4, «Using GRUB to Set Up the Boot Process».

6.44.2. Contents of GRUB

Installed programs:	grub-bin2h, grub-editenv, grub-install, grub-mkconfig, grub-mkdevicemap, grub-mkelfimage, grub-mkimage, grub-mkisofs, grub-mkpasswd-pbkdf2, grub-mkrelpath, grub-mkrescue, grub-probe, grub-reboot, grub-script-check, grub-set-default, grub-setup
Installed directories:	/usr/lib/grub, /etc/grub.d, /usr/share/grub

Short Descriptions

grub-bin2h	Converts a binary file to a C header
grub-editenv	A tool to edit the environment block
grub-install	Install GRUB on your drive
grub-mkconfig	Generate a grub config file
grub-mkdevicemap	Generate a device map file automatically
grub-mkelfimage	Make a bootable image of GRUB

grub-mkimage	Make a bootable image of GRUB
grub-mkisofs	Creates a bootable ISO image
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-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-script-check	Checks GRUB configuration script for syntax errors
grub-set-default	Sets the default boot entry for GRUB
grub-setup	Set up images to boot from a device

6.45. Gzip-1.4

The Gzip package contains programs for compressing and decompressing files.

Приблизительное время сборки:	less than 0.1 SBU
Требует свободного места на диске:	3.3 MB

6.45.1. Installation of Gzip

Prepare Gzip for compilation:

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

Compile the package:

```
make
```

To test the results, issue:

```
make check
```

Install the package:

```
make install
```

Move some programs that do not need to be on the root filesystem:

```
mv -v /bin/{gzexe,uncompress,zcmp,zdiff,zegrep} /usr/bin
mv -v /bin/{zfgrep,zforce,zgrep,zless,zmore,znew} /usr/bin
```

6.45.2. Contents of Gzip

Installed programs:	gunzip, gzexe, gzip, uncompress, zcat, zcmp, zdiff, zegrep, zfgrep, zforce, zgrep, zless, zmore, and znew

Short Descriptions

gunzip	Decompresses gzipped files
gzexe	Creates self-decompressing executable files
gzip	Compresses the given files using Lempel-Ziv (LZ77) coding
uncompress	Decompresses compressed files
zcat	Decompresses the given gzipped files to standard output
zcmp	Runs **cmp** on gzipped files
zdiff	Runs **diff** on gzipped files
zegrep	Runs **egrep** on gzipped files
zfgrep	Runs **fgrep** on gzipped files
zforce	Forces a `.gz` extension on all given files that are gzipped files, so that **gzip** will not compress them again; this can be useful when file names were truncated during a file transfer
zgrep	Runs **grep** on gzipped files

zless	Runs **less** on gzipped files
zmore	Runs **more** on gzipped files
znew	Re-compresses files from **compress** format to **gzip** format—`.Z` to `.gz`

6.46. IPRoute2-2.6.37

The IPRoute2 package contains programs for basic and advanced IPV4-based networking.

Приблизительное время сборки:	0.2 SBU
Требует свободного места на диске:	5.7 MB

6.46.1. Installation of IPRoute2

The **arpd** binary included in this package is dependent on Berkeley DB. Because **arpd** is not a very common requirement on a base Linux system, remove the dependency on Berkeley DB by applying the **sed** command below. If the **arpd** binary is needed, instructions for compiling Berkeley DB can be found in the BLFS Book at *http://www.linuxfromscratch.org/blfs/view/svn/server/databases.html#db*.

```
sed -i '/^TARGETS/s@arpd@@g' misc/Makefile
```

Fix a bug that causes the **ip route get** command to not produce any output:

```
sed -i '1289i\\tfilter.cloned = 2;' ip/iproute.c
```

Compile the package:

```
make DESTDIR=
```

The meaning of the make option:

DESTDIR=
: This ensures that the IPRoute2 binaries will install into the correct directory. By default, *DESTDIR* is set to `/usr`.

This package comes with a test suite, but due to assumptions it makes, it is not possible to reliably run these tests from within the chroot environment. If you wish to run these tests after booting into your new LFS system, ensure you select `/proc/config.gz` CONFIG_IKCONFIG_PROC ("General setup" -> "Enable access to .config through /proc/config.gz") support into your kernel then run 'make alltests' from the `testsuite/` subdirectory.

Install the package:

```
make DESTDIR= SBINDIR=/sbin MANDIR=/usr/share/man \
     DOCDIR=/usr/share/doc/iproute2-2.6.37 install
```

6.46.2. Contents of IPRoute2

Installed programs:	ctstat (link to lnstat), genl, ifcfg, ifstat, ip, lnstat, nstat, routef, routel, rtacct, rtmon, rtpr, rtstat (link to lnstat), ss, and tc
Installed directories:	/etc/iproute2, /lib/tc, /usr/share/doc/iproute2-2.6.37, /usr/lib/tc

Short Descriptions

ctstat Connection status utility

genl

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 continously 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

6.47. Kbd-1.15.2

The Kbd package contains key-table files and keyboard utilities.

Приблизительное время сборки:	less than 0.1 SBU
Требует свободного места на диске:	16.0 MB

6.47.1. Installation of Kbd

The behaviour of the Backspace and Delete keys is not consistent across the keymaps in the Kbd package. The following patch fixes this issue for i386 keymaps:

```
patch -Np1 -i ../kbd-1.15.2-backspace-1.patch
```

After patching, the Backspace key generates the character with code 127, and the Delete key generates a well-known escape sequence.

Prepare Kbd for compilation:

```
./configure --prefix=/usr --datadir=/lib/kbd
```

The meaning of the configure options:

`--datadir=/lib/kbd`
This option puts keyboard layout data in a directory that will always be on the root partition instead of the default `/usr/share/kbd`.

Compile the package:

```
make
```

This package does not come with a test suite.

Install the package:

```
make install
```

Замечание

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.

Some of the scripts in the LFS-Bootscripts package depend on **kbd_mode**, **loadkeys**, **openvt**, and **setfont**. As `/usr` may not be available during the early stages of booting, those binaries need to be on the root partition:

```
mv -v /usr/bin/{kbd_mode,loadkeys,openvt,setfont} /bin
```

If desired, install the documentation:

```
mkdir -v /usr/share/doc/kbd-1.15.2
cp -R -v doc/* \
         /usr/share/doc/kbd-1.15.2
```

6.47.2. Contents of Kbd

Installed programs:	chvt, deallocvt, dumpkeys, fgconsole, getkeycodes, kbd_mode, kbdrate, loadkeys, loadunimap, mapscrn, openvt, psfaddtable (link to psfxtable), psfgettable (link to psfxtable), psfstriptable (link to psfxtable), psfxtable, resizecons, setfont, setkeycodes, setleds, setmetamode, showconsolefont, showkey, unicode_start, and unicode_stop
Installed directory:	/lib/kbd

Short Descriptions

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
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	A link to **psfxtable**
psfgettable	A link to **psfxtable**
psfstriptable	A link to **psfxtable**
psfxtable	Handle Unicode character tables for console fonts
resizecons	Changes the kernel idea of the console size
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
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

6.48. Less-436

The Less package contains a text file viewer.

Приблизительное время сборки:	less than 0.1 SBU
Требует свободного места на диске:	2.9 MB

6.48.1. Installation of Less

Prepare Less for compilation:

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

The meaning of the configure options:

`--sysconfdir=/etc`
This option tells the programs created by the package to look in `/etc` for the configuration files.

Compile the package:

```
make
```

This package does not come with a test suite.

Install the package:

```
make install
```

6.48.2. Contents of Less

Installed programs:	less, lessecho, and lesskey

Short Descriptions

less	A file viewer or pager; it displays the contents of the given file, letting the user scroll, find strings, and jump to marks
lessecho	Needed to expand meta-characters, such as * and *?*, in filenames on Unix systems
lesskey	Used to specify the key bindings for **less**

6.49. Make-3.82

The Make package contains a program for compiling packages.

Приблизительное время сборки: 0.3 SBU
Требует свободного места на диске: 9.7 MB

6.49.1. Installation of Make

Prepare Make for compilation:

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

Compile the package:

```
make
```

To test the results, issue:

```
make check
```

Install the package:

```
make install
```

6.49.2. Contents of Make

Installed program: make

Short Descriptions

make Automatically determines which pieces of a package need to be (re)compiled and then issues the relevant commands

6.50. Xz-5.0.1

The Xz package contains programs for compressing and decompressing files. It provides capabilities for the lzma and the newer xz compression formats. Compressing text files with **xz** yields a better compression percentage than with the traditional **gzip** or **bzip2** commands.

Приблизительное время сборки:	0.4 SBU
Требует свободного места на диске:	13 MB

6.50.1. Installation of Xz

Prepare Xz for compilation with:

```
./configure --prefix=/usr --docdir=/usr/share/doc/xz-5.0.1
```

Compile the package:

```
make
```

To test the results, issue:

```
make check
```

Install the package:

```
make install
```

6.50.2. Contents of Xz

Installed programs:	lzcat (link to xz), lzcmp (link to xzdiff), lzdiff (link to xzdiff), bzdiff, lzegrep (link to xzgrep), lzfgrep (link to xzgrep), lz (link to xz), lzmadec, lzmainfo, lzmore (link to xzmore), unlzma (link to xz), xzcat (link to xz), xzcmp (link to xzdiff), xzdec, xzdiff, xzegrep (link to xzgrep), xzfgrep (link to xzgrep), xzgrep, xzless, xzmore
Installed libraries:	liblzma.{a,so}
Installed directories:	/usr/include/lzma and /usr/share/doc/xz-5.0.1

Short Descriptions

lzcat	Decompresses to standard output
lzcmp	Runs **cmp** on LZMA compressed files
lzdiff	Runs **diff** on LZMA compressed files
lzegrep	Runs **egrep** on LZMA compressed files files
lzfgrep	Runs **fgrep** on LZMA compressed files
lzgrep	Runs **grep** on LZMA compressed files
lzless	Runs **less** on LZMA compressed files
lzma	Compresses or decompresses files using the LZMA format
lzmadec	A small and fast decoder for LZMA compressed files

lzmainfo	Shows information stored in the LZMA compressed file header
lzmore	Runs **more** on LZMA compressed files
unlzma	Decompresses files using the LZMA format
unxz	Decompresses files using the XZ format
xz	Compresses or decompresses files using the XZ format
xzcat	Decompresses to standard output
xzcmp	Runs **cmp** on XZ compressed files
xzdec	A small and fast decoder for XZ compressed files
xzdiff	Runs **diff** on XZ compressed files
xzegrep	Runs **egrep** on XZ compressed files files
xzfgrep	Runs **fgrep** on XZ compressed files
xzgrep	Runs **grep** on XZ compressed files
xzless	Runs **less** on XZ compressed files
xzmore	Runs **more** on XZ compressed files
`liblzma*`	The library implementing lossless, block-sorting data compression, using the Lempel-Ziv-Markov chain algorithm

6.51. Man-DB-2.5.9

The Man-DB package contains programs for finding and viewing man pages.

Приблизительное время сборки:	0.4 SBU
Требует свободного места на диске:	22 MB

6.51.1. Installation of Man-DB

Prepare Man-DB for compilation:

```
./configure --prefix=/usr --libexecdir=/usr/lib \
    --docdir=/usr/share/doc/man-db-2.5.9 --sysconfdir=/etc --disable-setuid \
    --with-browser=/usr/bin/lynx --with-vgrind=/usr/bin/vgrind \
    --with-grap=/usr/bin/grap
```

The meaning of the configure options:

`--disable-setuid`
This disables making the **man** program setuid to user `man`.

`--with-...`
These three parameters are used to set some default programs. **lynx** is a text-based web browser (see BLFS for installation instructions), **vgrind** converts program sources to Groff input, and **grap** is useful for typesetting graphs in Groff documents. The **vgrind** and **grap** programs are not normally needed for viewing manual pages. They are not part of LFS or BLFS, but you should be able to install them yourself after finishing LFS if you wish to do so.

Compile the package:

```
make
```

To test the results, issue:

```
make -k check
```

Note that 2 tests are known to fail as they rely on warnings output from Groff, which changed slightly in Groff-1.21.

Install the package:

```
make install
```

6.51.2. Non-English Manual Pages in LFS

The following table shows the character set that Man-DB assumes manual pages installed under `/usr/share/man/<ll>` will be encoded with. In addition to this, Man-DB correctly determines if manual pages installed in that directory are UTF-8 encoded.

Таблица 6.1. Expected character encoding of legacy 8-bit manual pages

Language (code)	Encoding	Language (code)	Encoding
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		

Замечание

Manual pages in languages not in the list are not supported.

6.51.3. Contents of Man-DB

Installed programs: accessdb, apropos (link to whatis), catman, lexgrog, man, mandb, manpath, whatis, and zsoelim
Installed directories: /usr/lib/man-db, /usr/share/doc/man-db

Short Descriptions

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

zsoelim Reads files and replaces lines of the form *.so file* by the contents of the mentioned *file*

6.52. Module-Init-Tools-3.12

The Module-Init-Tools package contains programs for handling kernel modules in Linux kernels greater than or equal to version 2.5.47.

Приблизительное время сборки:	0.1 SBU
Требует свободного места на диске:	8.6 MB

6.52.1. Installation of Module-Init-Tools

To avoid a problem with regenerating the man pages when not needed, first rewrite a file that just points to another man page:

```
echo '.so man5/modprobe.conf.5' > modprobe.d.5
```

The test suite of this package is geared towards the needs of its Maintainer. The command **make check** builds a specially wrapped version of modprobe which is useless for normal operation. To run this (about 0.2 SBU), issue the following commands (note that the **make clean** command is required to clean up the source tree before recompiling for normal use):

```
./configure
make check
./tests/runtests
make clean
```

Prepare Module-Init-Tools for compilation:

```
./configure --prefix=/ --enable-zlib-dynamic --mandir=/usr/share/man
```

Compile the package:

```
make
```

Install the package:

```
make INSTALL=install install
```

The meaning of the make parameter:

`INSTALL=install`
Normally, **make install** will not install the binaries if they already exist. This option overrides that behavior by calling **install** instead of using the default wrapper script.

6.52.2. Contents of Module-Init-Tools

Installed programs:	depmod, insmod, insmod.static, lsmod, modinfo, modprobe, and rmmod

Short Descriptions

depmod	Creates a dependency file based on the symbols it finds in the existing set of modules; this dependency file is used by **modprobe** to automatically load the required modules

insmod	Installs a loadable module in the running kernel
insmod.static	A statically compiled version of **insmod**
lsmod	Lists currently loaded modules
modinfo	Examines an object file associated with a kernel module and displays any information that it can glean
modprobe	Uses a dependency file, created by **depmod**, to automatically load relevant modules
rmmod	Unloads modules from the running kernel

6.53. Patch-2.6.1

The Patch package contains a program for modifying or creating files by applying a «patch» file typically created by the **diff** program.

Приблизительное время сборки: less than 0.1 SBU
Требует свободного места на диске: 1.9 MB

6.53.1. Installation of Patch

Apply a patch to prevent the test suite from running a test that requires **ed**:

```
patch -Np1 -i ../patch-2.6.1-test_fix-1.patch
```

Prepare Patch for compilation:

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

Compile the package:

```
make
```

To test the results, issue:

```
make check
```

Install the package:

```
make install
```

6.53.2. Contents of Patch

Installed program: patch

Short Descriptions

patch Modifies files according to a patch file. A patch file is normally a difference listing created with the **diff** program. By applying these differences to the original files, **patch** creates the patched versions.

6.54. Psmisc-22.13

The Psmisc package contains programs for displaying information about running processes.

Приблизительное время сборки:	less than 0.1 SBU
Требует свободного места на диске:	2.5 MB

6.54.1. Installation of Psmisc

Prepare Psmisc for compilation:

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

Compile the package:

```
make
```

This package does not come with a test suite.

Install the package:

```
make install
```

Finally, move the **killall** and **fuser** programs to the location specified by the FHS:

```
mv -v /usr/bin/fuser /bin
mv -v /usr/bin/killall /bin
```

6.54.2. Contents of Psmisc

Installed programs:	fuser, killall, peekfd, prtstat, pstree, and pstree.x11 (link to pstree)

Short Descriptions

fuser	Reports the Process IDs (PIDs) of processes that use the given files or file systems
killall	Kills processes by name; it sends a signal to all processes running any of the given commands
peekfd	Peek at file descriptors of a running process, given its PID
prtstat	Prints information about a process
pstree	Displays running processes as a tree
pstree.x11	Same as **pstree**, except that it waits for confirmation before exiting

6.55. Shadow-4.1.4.3

The Shadow package contains programs for handling passwords in a secure way.

Приблизительное время сборки:	0.3 SBU
Требует свободного места на диске:	30 MB

6.55.1. Installation of Shadow

Замечание

If you would like to enforce the use of strong passwords, refer to *http://www.linuxfromscratch.org/blfs/view/svn/postlfs/cracklib.html* for installing CrackLib prior to building Shadow. Then add *--with-libcrack* to the **configure** command below.

Disable the installation of the **groups** program and its man pages, as Coreutils provides a better version:

```
sed -i 's/groups$(EXEEXT) //' src/Makefile.in
find man -name Makefile.in -exec sed -i 's/groups\.1 / /' {} \;
```

Fix an issue with the installation of Russian man pages:

```
sed -i 's/man_MANS = $(man_nopam) /man_MANS = /' man/ru/Makefile.in
```

Instead of using the default *crypt* method, use the more secure *SHA-512* method of password encryption, which also allows passwords longer than 8 characters. It is also necessary to change the obsolete `/var/spool/mail` location for user mailboxes that Shadow uses by default to the `/var/mail` location used currently:

```
sed -i -e 's@#ENCRYPT_METHOD DES@ENCRYPT_METHOD SHA512@' \
       -e 's@/var/spool/mail@/var/mail@' etc/login.defs
```

Замечание

If you chose to build Shadow with Cracklib support, run the following:

```
sed -i 's@DICTPATH.*@DICTPATH\t/lib/cracklib/pw_dict@' \
    etc/login.defs
```

Prepare Shadow for compilation:

```
./configure --sysconfdir=/etc
```

Compile the package:

```
make
```

This package does not come with a test suite.

Install the package:

```
make install
```

Move a misplaced program to its proper location:

```
mv -v /usr/bin/passwd /bin
```

6.55.2. Configuring Shadow

This package contains utilities to add, modify, and delete users and groups; set and change their passwords; and perform other administrative tasks. For a full explanation of what *password shadowing* means, see the `doc/HOWTO` file within the unpacked source tree. If using Shadow support, keep in mind that programs which need to verify passwords (display managers, FTP programs, pop3 daemons, etc.) must be Shadow-compliant. That is, they need to be able to work with shadowed passwords.

To enable shadowed passwords, run the following command:

```
pwconv
```

To enable shadowed group passwords, run:

```
grpconv
```

Shadow's stock configuration for the **useradd** utility has a few caveats that need some explanation. First, the default action for the **useradd** utility is to create the user and a group of the same name as the user. By default the user ID (UID) and group ID (GID) numbers will begin with 1000. This means if you don't pass parameters to **useradd**, each user will be a member of a unique group on the system. If this behaviour is undesireable, you'll need to pass the *-g* parameter to **useradd**. The default parameters are stored in the `/etc/default/useradd` file. You may need to modify two parameters in this file to suit your particular needs.

`/etc/default/useradd` Parameter Explanations

`GROUP=1000`

This parameter sets the beginning of the group numbers used in the /etc/group file. You can modify it to anything you desire. Note that **useradd** will never reuse a UID or GID. If the number identified in this parameter is used, it will use the next available number after this. Note also that if you don't have a group 1000 on your system the first time you use **useradd** without the *-g* parameter, you'll get a message displayed on the terminal that says: `useradd: unknown GID 1000`. You may disregard this message and group number 1000 will be used.

`CREATE_MAIL_SPOOL=yes`

This parameter causes **useradd** to create a mailbox file for the newly created user. **useradd** will make the group ownership of this file to the `mail` group with 0660 permissions. If you would prefer that these mailbox files are not created by **useradd**, issue the following command:

```
sed -i 's/yes/no/' /etc/default/useradd
```

6.55.3. Setting the root password

Choose a password for user *root* and set it by running:

```
passwd root
```

6.55.4. Contents of Shadow

Installed programs: chage, chfn, chgpasswd, chpasswd, chsh, expiry, faillog, gpasswd, groupadd, groupdel, groupmems, groupmod, grpck, grpconv, grpunconv, lastlog, login, logoutd, newgrp, newusers, nologin, passwd, pwck, pwconv, pwunconv, sg (link to newgrp), su, useradd, userdel, usermod, vigr (link to vipw), and vipw

Installed directory: /etc/default

Short Descriptions

chage Used to change the maximum number of days between obligatory password changes

chfn Used to change a user's full name and other information

chgpasswd Used to update group passwords in batch mode

chpasswd Used to update user passwords in batch mode

chsh Used to change a user's default login shell

expiry Checks and enforces the current password expiration policy

faillog Is used to examine the log of login failures, to set a maximum number of failures before an account is blocked, or to reset the failure count

gpasswd Is used to add and delete members and administrators to groups

groupadd Creates a group with the given name

groupdel Deletes the group with the given name

groupmems Allows a user to administer his/her own group membership list without the requirement of super user privileges.

groupmod Is used to modify the given group's name or GID

grpck Verifies the integrity of the group files `/etc/group` and `/etc/gshadow`

grpconv Creates or updates the shadow group file from the normal group file

grpunconv Updates `/etc/group` from `/etc/gshadow` and then deletes the latter

lastlog Reports the most recent login of all users or of a given user

login Is used by the system to let users sign on

logoutd Is a daemon used to enforce restrictions on log-on time and ports

newgrp Is used to change the current GID during a login session

newusers Is used to create or update an entire series of user accounts

nologin Displays a message that an account is not available. Designed to be used as the default shell for accounts that have been disabled

passwd Is used to change the password for a user or group account

pwck Verifies the integrity of the password files `/etc/passwd` and `/etc/shadow`

pwconv Creates or updates the shadow password file from the normal password file

pwunconv Updates `/etc/passwd` from `/etc/shadow` and then deletes the latter

sg Executes a given command while the user's GID is set to that of the given group

su Runs a shell with substitute user and group IDs

useradd	Creates a new user with the given name, or updates the default new-user information
userdel	Deletes the given user account
usermod	Is used to modify the given user's login name, User Identification (UID), shell, initial group, home directory, etc.
vigr	Edits the `/etc/group` or `/etc/gshadow` files
vipw	Edits the `/etc/passwd` or `/etc/shadow` files

6.56. Sysklogd-1.5

The Sysklogd package contains programs for logging system messages, such as those given by the kernel when unusual things happen.

Приблизительное время сборки:	less than 0.1 SBU
Требует свободного места на диске:	0.5 MB

6.56.1. Installation of Sysklogd

Compile the package:

```
make
```

This package does not come with a test suite.

Install the package:

```
make BINDIR=/sbin install
```

6.56.2. Configuring Sysklogd

Create a new `/etc/syslog.conf` file by running the following:

```
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
```

6.56.3. Contents of Sysklogd

Installed programs:	klogd and syslogd

Short Descriptions

klogd A system daemon for intercepting and logging kernel messages

syslogd Logs the messages that system programs offer for logging. Every logged message contains at least a date stamp and a hostname, and normally the program's name too, but that depends on how trusting the logging daemon is told to be

6.57. Sysvinit-2.88dsf

The Sysvinit package contains programs for controlling the startup, running, and shutdown of the system.

Приблизительное время сборки:	less than 0.1 SBU
Требует свободного места на диске:	1 MB

6.57.1. Installation of Sysvinit

When run-levels are changed (for example, when halting the system), **init** sends termination signals to those processes that **init** itself started and that should not be running in the new run-level. While doing this, **init** outputs messages like «Sending processes the TERM signal» which seem to imply that it is sending these signals to all currently running processes. To avoid this misinterpretation, modify the source so that these messages read like «Sending processes configured via /etc/inittab the TERM signal» instead:

```
sed -i 's@Sending processes@& configured via /etc/inittab@g' \
    src/init.c
```

A maintained version of the **wall** program was installed earlier by Util-linux. Suppress the installation of Sysvinit's version of this program and its man page:

```
sed -i -e 's/utmpdump wall/utmpdump/' \
       -e 's/mountpoint.1 wall.1/mountpoint.1/' src/Makefile
```

Compile the package:

```
make -C src
```

This package does not come with a test suite.

Install the package:

```
make -C src install
```

6.57.2. Configuring Sysvinit

Create a new file `/etc/inittab` by running the following:

```
cat > /etc/inittab << "EOF"
# Begin /etc/inittab

id:3:initdefault:

si::sysinit:/etc/rc.d/init.d/rc sysinit

l0:0:wait:/etc/rc.d/init.d/rc 0
l1:S1:wait:/etc/rc.d/init.d/rc 1
l2:2:wait:/etc/rc.d/init.d/rc 2
l3:3:wait:/etc/rc.d/init.d/rc 3
l4:4:wait:/etc/rc.d/init.d/rc 4
l5:5:wait:/etc/rc.d/init.d/rc 5
l6:6:wait:/etc/rc.d/init.d/rc 6

ca:12345:ctrlaltdel:/sbin/shutdown -t1 -a -r now

su:S016:once:/sbin/sulogin

1:2345:respawn:/sbin/agetty 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
```

6.57.3. Contents of Sysvinit

Installed programs:	bootlogd, fstab-decode, halt, init, killall5, last, lastb (link to last), mesg, mountpoint, pidof (link to killall5), poweroff (link to halt), reboot (link to halt), runlevel, shutdown, sulogin, telinit (link to init), and utmpdump

Short Descriptions

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 it is instructed to
killall5	Sends a signal to all processes, except the processes in its own session so it will not kill the shell running the script that called it

last	Shows which users last logged in (and out), searching back through the `/var/log/wtmp` file; it also shows system boots, shutdowns, and run-level changes
lastb	Shows the failed login attempts, as logged in `/var/log/btmp`
mesg	Controls whether other users can send messages to the current user's terminal
mountpoint	Checks if the directory is a mountpoint
pidof	Reports the PIDs of the given programs
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
sulogin	Allows `root` to log in; it is normally invoked by **init** when the system goes into single user mode
telinit	Tells **init** which run-level to change to
utmpdump	Displays the content of the given login file in a more user-friendly format

6.58. Tar-1.25

The Tar package contains an archiving program.

Приблизительное время сборки:	1.9 SBU
Требует свободного места на диске:	21.2 MB

6.58.1. Installation of Tar

Prepare Tar for compilation:

```
FORCE_UNSAFE_CONFIGURE=1 ./configure --prefix=/usr \
    --bindir=/bin --libexecdir=/usr/sbin
```

The meaning of the configure options:

`FORCE_UNSAFE_CONFIGURE=1`
This forces the test for `mknod` to be run as root. It is generally considered dangerous to run this test as the root user, but as it is being run on an only partially built system, overriding it is OK.

Compile the package:

```
make
```

To test the results (about 1 SBU), issue:

```
make check
```

Install the package:

```
make install
make -C doc install-html docdir=/usr/share/doc/tar-1.25
```

6.58.2. Contents of Tar

Installed programs:	rmt and tar

Short Descriptions

rmt Remotely manipulates a magnetic tape drive through an interprocess communication connection

tar Creates, extracts files from, and lists the contents of archives, also known as tarballs

6.59. Texinfo-4.13a

The Texinfo package contains programs for reading, writing, and converting info pages.

Приблизительное время сборки:	0.3 SBU
Требует свободного места на диске:	21 MB

6.59.1. Installation of Texinfo

Prepare Texinfo for compilation:

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

Compile the package:

```
make
```

To test the results, issue:

```
make check
```

Install the package:

```
make install
```

Optionally, install the components belonging in a TeX installation:

```
make TEXMF=/usr/share/texmf install-tex
```

The meaning of the make parameter:

TEXMF=/usr/share/texmf
The TEXMF makefile variable holds the location of the root of the TeX tree if, for example, a TeX package will be installed later.

The Info documentation system uses a plain text file to hold its list of menu entries. The file is located at /usr/share/info/dir. Unfortunately, due to occasional problems in the Makefiles of various packages, it can sometimes get out of sync with the info pages installed on the system. If the /usr/share/info/dir file ever needs to be recreated, the following optional commands will accomplish the task:

```
cd /usr/share/info
rm -v dir
for f in *
do install-info $f dir 2>/dev/null
done
```

6.59.2. Contents of Texinfo

Installed programs:	info, infokey, install-info, makeinfo, pdftexi2dvi, texi2dvi, texi2pdf, and texindex
Installed directory:	/usr/share/texinfo

Short Descriptions

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**.
infokey	Compiles a source file containing Info customizations into a binary format
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
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

6.60. Udev-166

The Udev package contains programs for dynamic creation of device nodes.

Приблизительное время сборки:	0.2 SBU
Требует свободного места на диске:	9.3 MB plus 37 MB for testfiles

6.60.1. Installation of Udev

The udev-config tarball contains LFS-specific files used to configure Udev. Unpack it into the Udev source directory:

```
tar -xvf ../udev-config-20100128.tar.bz2
```

The udev-testfiles tarball contains files needed to test udev. The file expands to an apparent size of about 37MB but the actual disk usage is less than 7MB.

```
tar -xvf ../udev-166-testfiles.tar.bz2 --strip-components=1
```

Create some devices and directories that Udev cannot handle due to them being required very early in the boot process, or by Udev itself:

```
install -dv /lib/{firmware,udev/devices/{pts,shm}}
mknod -m0666 /lib/udev/devices/null c 1 3
```

Prepare the package for compilation:

```
./configure --prefix=/usr \
    --sysconfdir=/etc --sbindir=/sbin \
    --with-rootlibdir=/lib --libexecdir=/lib/udev \
    --disable-extras --disable-introspection
```

The meaning of the new configure options

`--with-rootlibdir=/lib`
This controls where the `libudev` library is installed. The library needs to be in `/lib` because it's used by Udev at boot time, before `/usr` might be available, and the default --rootlibdir is `/usr/lib`.

`--libexecdir=/lib/udev`
This controls where Udev-internal rules and helper programs are installed.

`--disable-extras`
This option prevents Udev from installing helper programs and other extras which require more external libraries. These libraries are not part of the base LFS system. See the Udev `README` file for more information.

`--disable-introspection`
This option prevents Udev's instrospection feature, which requires packages not installed as part of the base LFS system. See the Udev `README` file for more information.

Compile the package:

```
make
```

Test the package.

```
make check
```

Install the package:

```
make install
```

Remove an empty documentation directory:

```
rmdir -v /usr/share/doc/udev
```

Now install the LFS-specific custom rules files:

```
cd udev-config-20100128
make install
```

Install the documentation that explains the LFS-specific rules files:

```
make install-doc
```

6.60.2. Contents of Udev

Installed programs:	ata_id, cdrom_id, collect, create_floppy_devices, edd_id, firmware.sh, fstab_import, path_id, scsi_id, udevadm, udevd, usb_id, write_cd_rules, and write_net_rules
Installed libraries:	libudev.{a,so}
Installed directories:	/etc/udev, /lib/udev, /lib/firmware

Short Descriptions

ata_id	Provides Udev with a unique string and additional information (uuid, label) for an ATA drive
cdrom_id	Provides Udev with the capabilities of a CD-ROM or DVD-ROM drive
collect	Given an ID for the current uevent and a list of IDs (for all target uevents), registers the current ID and indicates whether all target IDs have been registered
create_floppy_devices	Creates all possible floppy devices based on the CMOS type
edd_id	Provides Udev with the EDD ID for a BIOS disk drive
firmware.sh	Uploads firmware to devices
fstab_import	Finds an entry in `/etc/fstab` that matches the current device, and provides its information to Udev
path_id	Provides the shortest possible unique hardware path to a device
scsi_id	Provides Udev with a unique SCSI identifier based on the data returned from sending a SCSI INQUIRY command to the specified device
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 and runs the configured external programs in response to these uevents
usb_id	Provides Udev with information about USB devices
write_cd_rules	A script which generates Udev rules to provide stable names for optical drives (see also Раздел 7.10, «Создание собственных ссылок на устройства»)
write_net_rules	A script which generates rules to provide stable names for network interfaces (see also Раздел 7.13, «Configuring the network Script»)
`libudev`	A library interface to udev device information
`/etc/udev`	Contains Udev configuration files, device permissions, and rules for device naming

6.61. Vim-7.3

The Vim package contains a powerful text editor.

Приблизительное время сборки:	1.0 SBU
Требует свободного места на диске:	87 MB

Alternatives to Vim

If you prefer another editor—such as Emacs, Joe, or Nano—please refer to *http://www.linuxfromscratch.org/blfs/view/svn/postlfs/editors.html* for suggested installation instructions.

6.61.1. Installation of Vim

First, change the default location of the `vimrc` configuration file to `/etc`:

```
echo '#define SYS_VIMRC_FILE "/etc/vimrc"' >> src/feature.h
```

Now prepare Vim for compilation:

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

The meaning of the configure options:

`--enable-multibyte`
This switch enables support for editing files in multibyte character encodings. This is needed if using a locale with a multibyte character set. This switch is also helpful to be able to edit text files initially created in Linux distributions like Fedora that use UTF-8 as a default character set.

Compile the package:

```
make
```

To test the results, issue:

```
make test
```

However, this test suite outputs a lot of binary data to the screen, which can cause issues with the settings of the current terminal. This can be resolved by redirecting the output to a log file.

Install the package:

```
make install
```

Many users are used to using **vi** instead of **vim**. To allow execution of **vim** when users habitually enter **vi**, create a symlink for both the binary and the man page in the provided languages:

```
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
```

By default, Vim's documentation is installed in `/usr/share/vim`. The following symlink allows the documentation to be accessed via `/usr/share/doc/vim-7.3`, making it consistent with the location of documentation for other packages:

```
ln -sv ../vim/vim73/doc /usr/share/doc/vim-7.3
```

If an X Window System is going to be installed on the LFS system, it may be necessary to recompile Vim after installing X. Vim comes with a GUI version of the editor that requires X and some additional libraries to be installed. For more information on this process, refer to the Vim documentation and the Vim installation page in the BLFS book at *http://www.linuxfromscratch.org/blfs/view/svn/postlfs/editors.html#postlfs-editors-vim*.

6.61.2. Configuring Vim

By default, **vim** runs in vi-incompatible mode. This may be new to users who have used other editors in the past. The «nocompatible» setting is included below to highlight the fact that a new behavior is being used. It also reminds those who would change to «compatible» mode that it should be the first setting in the configuration file. This is necessary because it changes other settings, and overrides must come after this setting. Create a default **vim** configuration file by running the following:

```
cat > /etc/vimrc << "EOF"
" Begin /etc/vimrc

set nocompatible
set backspace=2
syntax on
if (&term == "iterm") || (&term == "putty")
  set background=dark
endif

" End /etc/vimrc
EOF
```

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. 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 following command:

```
vim -c ':options'
```

Замечание

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/vim73/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.

6.61.3. Contents of Vim

Installed programs:	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
Installed directory:	/usr/share/vim

Short Descriptions

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 show 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

6.62. About Debugging Symbols

Most programs and libraries are, by default, compiled with debugging symbols included (with **gcc**'s *-g* option). This means that when debugging a program or library that was compiled with debugging information included, the debugger can provide not only memory addresses, but also the names of the routines and variables.

However, the inclusion of these debugging symbols enlarges a program or library significantly. The following is an example of the amount of space these symbols occupy:

- A **bash** binary with debugging symbols: 1200 KB
- A **bash** binary without debugging symbols: 480 KB
- Glibc and GCC files (`/lib` and `/usr/lib`) with debugging symbols: 87 MB
- Glibc and GCC files without debugging symbols: 16 MB

Sizes may vary depending on which compiler and C library were used, but when comparing programs with and without debugging symbols, the difference will usually be a factor between two and five.

Because most users will never use a debugger on their system software, a lot of disk space can be regained by removing these symbols. The next section shows how to strip all debugging symbols from the programs and libraries.

6.63. Stripping Again

If the intended user is not a programmer and does not plan to do any debugging on the system software, the system size can be decreased by about 90 MB by removing the debugging symbols from binaries and libraries. This causes no inconvenience other than not being able to debug the software fully anymore.

Most people who use the command mentioned below do not experience any difficulties. However, it is easy to make a typo and render the new system unusable, so before running the **strip** command, it is a good idea to make a backup of the LFS system in its current state.

Before performing the stripping, take special care to ensure that none of the binaries that are about to be stripped are running. If unsure whether the user entered chroot with the command given in Раздел 6.4, «Entering the Chroot Environment,» first exit from chroot:

```
logout
```

Then reenter it with:

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

Now the binaries and libraries can be safely stripped:

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

A large number of files will be reported as having their file format not recognized. These warnings can be safely ignored. These warnings indicate that those files are scripts instead of binaries.

If disk space is very tight, the *--strip-all* option can be used on the binaries in /{,usr/}{bin,sbin} to gain several more megabytes. Do not use this option on libraries—they will be destroyed.

6.64. Cleaning Up

From now on, when reentering the chroot environment after exiting, use the following modified chroot command:

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

The reason for this is that the programs in /tools are no longer needed. Since they are no longer needed you can delete the /tools directory if so desired.

Замечание

Removing /tools will also remove the temporary copies of Tcl, Expect, and DejaGNU which were used for running the toolchain tests. If you need these programs later on, they will need to be recompiled and re-installed. The BLFS book has instructions for this (see *http://www.linuxfromscratch.org/blfs/*).

If the virtual kernel file systems have been unmounted, either manually or through a reboot, ensure that the virtual kernel file systems are mounted when reentering the chroot. This process was explained in Раздел 6.2.2, «Mounting and Populating /dev» and Раздел 6.2.3, «Mounting Virtual Kernel File Systems».

Глава 7. Установка загрузочных скриптов

7.1. Вступление

Эта глава описывает установку и настройку пакета LFS-Bootscripts. Большая часть этих скриптов будет работать без изменений, но все же необходимо создать несколько дополнительных конфигурационных файлов, поскольку они будут предоставлять информацию об оборудовании.

Загрузочные скрипты в стиле System V выбраны в этой книге по причине их наибольшей распространенности. Совет, описывающий установку загрузочных скриптов в стиле BSD, можно прочесть здесь: *http://www.linuxfromscratch.org/hints/downloads/files/bsd-init.txt*. Поиск в списках рассылки LFS по слову «depinit» позволит Вам ознакомиться с дополнительными вариантами.

Если Вы используете альтернативный стиль загрузочных скриптов, пропустите эту главу и переходите сразу к Chapter 8.

7.2. LFS-Bootscripts-20100627

Пакет LFS-Bootscripts содержит набор скриптов для запуска/остановки системных служб при загрузке или выключении LFS-системы.

Приблизительное время сборки: less than 0.1 SBU
Требует свободного места на диске: 468 KB

7.2.1. Установка LFS-Bootscripts

Установим пакет:

```
make install
```

7.2.2. Содержимое LFS-Bootscripts

Установленные скрипты: checkfs, cleanfs, console, consolelog, functions, halt, ifdown, ifup, localnet, modules, mountfs, mountkernfs, network, rc, reboot, sendsignals, setclock, static, swap, sysctl, sysklogd, template, udev и udev_retry

Краткое описание

checkfs Проверяет состояние файловых систем перед их монтированием (исключая журналы файловых систем и сетевые файловые системы)

cleanfs Удаляет файлы, которые не должны сохраняться между перезагрузками, например файлы в директориях `/var/run/` и `/var/lock/`; создает заново директорию `/var/run/utmp` и удаляет файлы `/etc/nologin`, `/fastboot` и `/forcefsck`, если они существуют

console Загружает таблицу символов клавиатуры, соответствующую выбранной раскладке; также устанавливает шрифт для консоли

consolelog Устанавливает, насколько подробно должны выводиться сообщения ядра на консоль

functions Содержит общие для всех скриптов функции, такие как проверка статуса и ошибок

halt Останавливает систему

ifdown Помогает скрипту network остановить сетевые устройства

ifup Помогает скрипту network запустить сетевые устройства

localnet Устанавливает имя системы и настраивает сетевое устройство "loopback"

modules Загружает модули ядра, перечисленные в файле `/etc/sysconfig/modules`, используя аргументы, также задаваемые в нем

mountfs Монтирует все файловые системы, за исключением сетевых и имеющих опцию *noauto*

mountkernfs Монтирует виртуальные системы ядра, например `proc`

network Задействует сетевые карты, поднимает сетевые интерфейсы и устанавливает шлюз по умолчанию (где возможно)

rc Основной скрипт контроля уровня запуска; он отвечает за последовательный запуск всех остальных скриптов в порядке, определенном символическими ссылками

reboot Перезагружает систему

sendsignals Перед перезагрузкой или выключением системы сначала посылает всем процессам сигналы, требующие их завершения, а затем уничтожает оставшиеся процессы

setclock Устанавливает на часах ядра локальное время, если на аппаратных часах время не в UTC

static Предоставляет необходимую функциональность для присвоения статического IP-адреса сетевому интерфейсу

swap Включает/отключает файлы и разделы подкачки

sysctl Считывает конфигурацию системы из файла `/etc/sysctl.conf`, если он существует, и передает ее ядру

sysklogd Запускает/останавливает демонов журналирования системы и ядра

template Шаблон для создания своих скриптов

udev Подготавливает директорию `/dev` и запускает Udev

udev_retry Пытается заново выполнить неудавшиеся события udev и копирует созданные файлы правил из директории `/dev/.udev` в `/etc/udev/rules.d`, если необходимо

7.3. Как работают загрузочные скрипты?

Linux использует специальную технологию, называемую SysVinit, которая базируется на понятии "уровней" - *runlevels*. В разных дистрибутивах некоторые особенности могут отличаться, поэтому нельзя просто взять скрипты из другой системы и установить их в LFS. LFS следует своим собственным путем, но при этом постоянно оглядываясь на стандарты.

SysVinit (в дальнейшем мы будем ссылаться на нее как на просто «init») работает с использованием механизма уровней запуска. Существуют 7 (от 0 до 6) уровней (на самом деле их несколько больше, но дополнительные уровни 7-9 обычно не используются. Обратитесь к `init(8)` за более подробной информацией), каждый из которых описывает действия, которые компьютер должен выполнить при входе на этот уровень. По умолчанию используется уровень 3. Ниже приведено краткое описание разных уровней в соответствии со стандартами:

```
0: выключение компьютера
1: однопользовательский режим
2: многопользовательский режим без поддержки сети
3: многопользовательский режим с поддержкой сети
4: зарезервирован, обычно соответствует 3
5: то же, что и 4, обычно используется для графического входа в систему (с помощью x
6: перезагрузка компьютера
```

Для смены текущего уровня используется команда **init *<число>***, где *<число>* - уровень, на который необходимо перейти. Например, для перезагрузки компьютера пользователь может выполнить команду **init 6**, которая является псевдонимом для **reboot**. Аналогично, **init 0** - псевдоним для команды **halt**.

В директории `/etc/rc.d` расположены несколько поддиректорий вида `rc?.d` (где ? - номер уровня) и директория `rcsysinit.d`, которые содержат символические ссылки на скрипты. Имена некоторых начинаются с *K*, других с *S*, и после каждой из этих букв следует двузначное число. K обозначает остановку сервиса (kill), а S означает запуск (start). Числа определяют порядок запуска скриптов, от 00 до 99 — меньшее число соответствует более раннему запуску. Когда **init** переключается на другой уровень, службы запускаются или останавливаются в соответствии с выбранным уровнем.

Сами скрипты расположены в директории `/etc/rc.d/init.d`. Они и выполняют все действия, и все символические ссылки указывают на них. Ссылки запуска сервиса и ссылки его остановки указывают на один и тот же скрипт в `/etc/rc.d/init.d`. Дело в том, что скрипты вызываются с различными параметрами, такими как *`start`*, *`stop`*, *`restart`*, *`reload`* и *`status`*. Если ссылка начинается с K, скрипт запускается с параметром *`stop`*. Если же ссылка начинается с S, соответствующему скрипту передается параметр *`start`* .

Из этой схемы существует одно исключение. Ссылки, начинающиеся с *S* и расположенные в директориях `rc0.d` и `rc6.d` ничего не запустят. Они будут вызваны с параметром *`stop`* на случай, если эти службы запущены и их нужно остановить. И в самом деле, если пользователь выключает или перезагружает систему, нет нужды ничего запускать.

Ниже - список того, какие аргументы каким действиям соответствуют:

`start`
  Служба запускается.

stop
Служба останавливается.

restart
Служба останавливается и запускается снова.

reload
Необходимо обновить настройки службы. Используется после изменения файла конфигурации сервиса, когда его не нужно перезапускать.

status
Сообщает, запущен ли сервис и если да, то с каким PID.

Вы совершенно свободно можете изменять эти скрипты (в конце концов, это Ваша собственная LFS-система). Приведенные здесь файлы являются просто примером того, какими могут быть загрузочные скрипты.

7.4. Настройка скрипта setclock

Скрипт **setclock** считывает время с аппаратных часов, также известных как часы BIOS или CMOS (Complementary Metal Oxide Semiconductor). Если на аппаратных часах установлено UTC (всемирное скоординированное время), этот скрипт преобразует полученное значение в локальное время, используя файл `/etc/localtime` (который сообщает программе **hwclock**, в какой временной зоне находится пользователь). Невозможно определить, установлено на аппаратных часах UTC или же локальное время, поэтому необходимо указать это вручную.

Скрипт **setclock** запускается системой udev при загрузке, когда ядро определяет возможности аппаратного обеспечения. Также его можно запустить вручную с параметром stop, чтобы сохранить системное время в аппаратные часы CMOS.

Если Вы не можете вспомнить, установлено на аппаратных часах UTC или локальное время, запустите команду **hwclock --localtime --show**. Она отобразит текущее время в соответствии с аппаратными часами. Если оно совпадает с тем, что показывают Ваши настенные/наручные часы, значит на часах CMOS установлено локальное время. Если вывод **hwclock** не совпадает с локальным временем, скорее всего, это UTC. Проверьте это, добавляя или вычитая правильное смещение к времени, выводимому **hwclock**. Например, если Вы живете во временной зоне MST, также известной как GMT -0700, добавьте семь часов к локальному времени.

Измените значение переменной `UTC` ниже на *`0`* (ноль), если на аппаратных часах установлено *не* UTC.

Создайте новый файл `/etc/sysconfig/clock`, запустив:

```
cat > /etc/sysconfig/clock << "EOF"
# Начало /etc/sysconfig/clock

UTC=1

# Добавьте в эту переменную опции, которые хотите передавать hwclock,
# например, тип аппаратных часов на архитектуре Alpha.
CLOCKPARAMS=

# Конец /etc/sysconfig/clock
EOF
```

Хороший совет, объясняющий, как обращаться с временем в LFS, доступен тут: *http://www.linuxfromscratch.org/hints/downloads/files/time.txt*. Он рассказывает о временных зонах, UTC и переменной окружения TZ.

7.5. Настройка консоли Linux

Эта секция описывает настройку скриптов **console** и **consolelog**, которые устанавливают раскладку клавиатуры, шрифт консоли и уровень подробности информации, выводимой ядром на консоль. Если Вы не планируете использовать символы, не соответствующие стандарту ASCII (например, знак копирайта, символы фунта и евро), и собираетесь печатать только в английской раскладке, то можете пропустить большую часть секции. Без файла конфигурации скрипт **console** не будет ничего делать.

Скрипты **console** и **consolelog** считывают конфигурацию из файла `/etc/sysconfig/console`. Решите для себя, какую раскладку клавиатуры и какой шрифт намерены использовать. Различные HOWTO для многих языков можно найти здесь: *http://www.tldp.org/HOWTO/HOWTO-INDEX/other-lang.html*. Если Вы все еще сомневаетесь, посмотрите список доступных раскладок и шрифтов в директории `/lib/kbd`. Прочтите страницы руководства `loadkeys(1)` и `setfont(8)`, чтобы узнать необходимые аргументы для этих программ.

Файл `/etc/sysconfig/console` должен содержать строки в формате ПЕРЕМЕННАЯ="значение". Допустимы следующие переменные:

LOGLEVEL
: Эта переменная задает уровень подробности сообщений, посылаемых ядром на системную консоль. Значение этой переменной передается в качестве аргумента утилите **dmesg**. Допустимы уровни от "1" (нет сообщений) до "8". По умолчанию "7".

KEYMAP
: Указывает аргументы для программы **loadkeys**, обычно имя раскладки, например, «es». Если эта переменная не установлена, загрузочные скрипты не запустят **loadkeys** и будет использоваться раскладка по умолчанию.

KEYMAP_CORRECTIONS
: Эта (крайне редко используемая) переменная задает аргументы для второго вызова программы **loadkeys**. Она полезна, если стандартная раскладка Вас не совсем удовлетворяет и Вы хотите немного ее подправить. Например, чтобы добавить символ евро в раскладку, которая его не содержит, присвойте этой переменной значение «euro2».

FONT
: Задает аргументы для утилиты **setfont**. Обычно, она включает имя шрифта, «-m» и затем имя карты символов. Например, чтобы загрузить шрифт «lat1-16» вместе с картой символов «8859-1» (как принято в США), присвойте переменной значение «lat1-16 -m 8859-1». В режиме UTF-8, ядро использует карту символов для преобразования 8-битных кодов нажатых клавиш из раскладке в UTF-8 и аргумент параметра "-m" должен указывать кодировку кодов клавиш в раскладке.

UNICODE
: Присвойте этой переменной значение «1», «yes» или «true», чтобы переключить консоль в режим UTF-8. Это полезно при использовании локали, основанной на UTF-8, и не рекомендуется в иных случаях.

LEGACY_CHARSET

Для многих раскладок клавиатуры в пакете Kbd не существует готового Unicode-варианта. Скрипт **console** будет на лету конвертировать имеющуюся раскладку в UTF-8, если привоить этой переменной имя доступной не-UTF-8 раскладки.

Несколько примеров:

- Для не-Unicode настройки необходимы только переменные KEYMAP и FONT. Например, для польских пользователей может подойти такой вариант:

```
cat > /etc/sysconfig/console << "EOF"
# Начало /etc/sysconfig/console

KEYMAP="pl2"
FONT="lat2a-16 -m 8859-2"

# Конец /etc/sysconfig/console
EOF
```

- Как упоминалось выше, иногда бывает необходимо подкорректировать раскладку. Следующий пример добавляет символ евро к немецкой раскладке:

```
cat > /etc/sysconfig/console << "EOF"
# Начало /etc/sysconfig/console

KEYMAP="de-latin1"
KEYMAP_CORRECTIONS="euro2"
FONT="lat0-16 -m 8859-15"

# Конец /etc/sysconfig/console
EOF
```

- Следующий пример - Болгарский язык в режиме Unicode, поскольку для этого языка существует UTF-8 раскладка:

```
cat > /etc/sysconfig/console << "EOF"
# Начало /etc/sysconfig/console

UNICODE="1"
KEYMAP="bg_bds-utf8"
FONT="LatArCyrHeb-16"

# Конец /etc/sysconfig/console
EOF
```

- Из-за использования 512-символьного шрифта LatArCyrHeb-16 в предыдущем примере, Вы не сможете использовать яркие цвета в консоли Linux без применения буфера кадров. Если Вы хотите использовать яркие цвета без буфера

кадров и готовы прожить без символов, не относящихся к Вашему языку, Вы можете использовать специфичный для вашего языка 256-символьный шрифт, как показано ниже:

```
cat > /etc/sysconfig/console << "EOF"
# Начало /etc/sysconfig/console

UNICODE="1"
KEYMAP="bg_bds-utf8"
FONT="cyr-sun16"

# Конец /etc/sysconfig/console
EOF
```

- Следующий пример демонстрирует автопреобразование раскладки из UTF-85 в UTF-8 и включает "мертвые" клавиши в режиме Unicode:

```
cat > /etc/sysconfig/console << "EOF"
# Начало /etc/sysconfig/console

UNICODE="1"
KEYMAP="de-latin1"
KEYMAP_CORRECTIONS="euro2"
LEGACY_CHARSET="iso-8859-15"
FONT="LatArCyrHeb-16 -m 8859-15"

# Конец /etc/sysconfig/console
EOF
```

- Некоторые раскладки включают в себя "мертвые" клавиши (то есть клавиши, нажатие которых само по себе не приводит к появлению на экране символа, но которые влияют на символ, генерируемый следующей клавишей) или определяют слияние символов (например: «нажмите Ctrl+. A E, чтобы получить Æ» в раскладке по умолчанию). Linux-2.6.37 правильно интерпретирует "мертвые" клавиши и слияния, только когда исходные символы имеют 8-битные коды. Эта особенность не влияет на раскладки для европейских языков, поскольку в них "сливаются" два ASCII-символа или добавляются подчеркивания к неподчеркнутым ASCII-символам. Однако, в режиме UTF-8 могут быть проблемы, например, для греческого языка, когда необходимо подчеркнуть символ «alpha». Решением в этой ситуации будет отказ от использования UTF-8 или установка графической системы X Window, не имеющих подобных ограничений.
- Для китайского, японского, корейского и некоторых других языков невозможно насторить консоль Linux так, чтобы она отображала все необходимые символы. Пользователи, которым требуются эти языки, должны установить систему X Window, шрифты, покрывающие необходимый диапазон символов, и правильный метод ввода (например, SCIM, он поддерживает большое число разнообразных языков).

Замечание

Файл `/etc/sysconfig/console` управляет только локализацией текстовой консоли Linux. Он никак не влияет на настройки раскладки клавиатуры и шрифтов в системе X Window, в сессиях SSH или на последовательном терминале. В этих ситуациях ограничения, описанные в двух расположенных выше абзацах, не применяются.

7.6. Настройка скрипта sysklogd

Скрипт `sysklogd` запускает программу **syslogd** с опцией *`-m 0`*. Этот параметр отключает периодическую (по умолчанию - каждые 20 минут) запись временных меток в файлы журналов, производимую **syslogd**. Если Вам необходимо включить периодическую запись временных меток, отредактируйте скрипт `sysklogd` и внесите соответствующие изменения. Обратитесь к **man syslogd** за дополнительной информацией.

7.7. Создание файла /etc/inputrc

В файле `inputrc` можно настроить параметры клавиатуры для специфических случаев. Этот файл используется Readline — библиотекой ввода — и считывается при запуске Bash и большей частью других оболочек.

Большинство людей не нуждаются в специальных настройках клавиатуры для каждого пользователя, поэтому команда ниже создаст общесистемный `/etc/inputrc`, используемый всеми. Если позже Вы решите переназначить для одного из пользователей умолчания, Вы можете создать файл `.inputrc` в домашней папке пользователя и указать в нем измененные настройки.

За более подробной информацией по редактированию файла `inputrc` , прочтите секцию *Readline Init File* на странице **info bash**. Также хорошим источником информации является **info readline**.

Ниже - обобщенный пример файла inputrc с комментариями к каждой опции. Заметьте, что комментарии не могут быть на одной строке с командами. Создайте файл следующей командой:

```
cat > /etc/inputrc << "EOF"
# Начало /etc/inputrc
# Изменен Chris Lynn <roryo@roryo.dynup.net>

# Разрешить перенос приглашения оболочки на следующую строку
set horizontal-scroll-mode Off

# Разрешить 8-битный ввод
set meta-flag On
set input-meta On

# Отключить обрезание восьмого бита в вводимых символах
set convert-meta Off

# Выводить на экран все восемь бит, не обрезая
set output-meta On

# Звуковой сигнал - none, visible или audible, соответственно никакого, видимы
set bell-style none

# Нижеследующие команды привязывают escape-последовательности (первый аргумент
# к специфичным для Readline функциям (второй аргумент)
"\eOd": backward-word
"\eOc": forward-word

# для консоли linux
"\e[1~": beginning-of-line
"\e[4~": end-of-line
"\e[5~": beginning-of-history
"\e[6~": end-of-history
"\e[3~": delete-char
"\e[2~": quoted-insert

# для xterm
"\eOH": beginning-of-line
"\eOF": end-of-line

# для Konsole
"\e[H": beginning-of-line
"\e[F": end-of-line

# Конец /etc/inputrc
EOF
```

7.8. Файлы конфигурации оболочки Bash

Программа оболочки **/bin/bash** (далее просто «оболочка») использует несколько файлов для создания праильного рабочего окружения. Каждый файл имеет собственное значение и может оказывать разное влияние на интерактивное окружение и окружение входа. Файлы в директории `/etc` содержат глобальные настройки. Если эквивалентный файл существует в домашней папке, он может переопределить глобальные значения.

Интерактивная оболочка входа запускается после успешного входа в систему программой **/bin/login**, считывающей файл `/etc/passwd`. Интерактивная не входная оболочка запускается из командной строки (например, `[prompt]$` **/bin/bash**). Неинтерактивная оболочка, как правило, присутствует при выполнении скрипта. Она не является интерактивной потому, что обрабатывает скрипт и не ждет ввода пользователя между командами.

За более подробной информацией обратитесь к странице **info bash**, секции *Bash Startup Files and Interactive Shells*.

Файлы `/etc/profile` и `~/.bash_profile` считываются, когда **bash** запускается как интерактивная оболочка входа.

Глобальный файл `/etc/profile`, приведенный ниже, устанавливает некоторые переменные окружения, необходимые для поддержки языка. Правильные их значения гарантируют:

- Вывод программ будет переводиться на родной для пользователя язык
- Правильную классификацию символов на буквы, цифры и другие типы. Это необходимо для верного распознавания **bash** не-ASCII символов в командной строке при неанглийских локалях
- Корректный порядок алфавитной сортировки для выбранной страны
- Правильный размер бумаги по умолчанию
- Правильное форматирование валюты, времени и даты

Замените *<ll>* на двузначный код Вашего языка (например, «en») и *<CC>* на двузначный код Вашей страны (например, «GB»). *<charmap>* необходимо заменить на карту символов для выбранной локали, например UTF-8. Можно также добавить необязательные модификаторы, такие как «@euro».

Список всех локалей, поддерживаемых Glibc, можно получить командой:

```
locale -a
```

Карты символов могут иметь несколько псевдонимов, например, «UTF-8» также обозначается как «iso8859-1» и «iso88591». Некоторые приложения не могут корректно обрабатывать различные варианты (к примеру, требуют чтобы «UTF-8» обозначался как «UTF-8», а не «utf8»), поэтому в большинстве случаев лучше всего использовать стандартное имя локали. Чтобы определить его, запустите следующую команду, где *<localename>* - вывод, полученный от команды **locale -a** для Вашей локали («en_GB.iso88591» в нашем примере):

```
LC_ALL=<localename> locale charmap
```

Для локали «en_GB.iso88591» будет выдано:

```
UTF-8
```

В результате окончательным именем локали будет «en_GB.UTF-8». Необходимо убедиться, что эта локаль является корректной и рабочей перед тем, как добавить ее в файлы конфигурации Bash:

```
LC_ALL=<locale name> locale language
LC_ALL=<locale name> locale charmap
LC_ALL=<locale name> locale int_curr_symbol
LC_ALL=<locale name> locale int_prefix
```

Эти команды выведут имя языка, кодировку символов, используемую локалью, обозначение валюты и префикс телефонного номера для выбранной страны. Если хотя бы одна из команд завершается неудачно и выводит нечто похожее на сообщение ниже, это значит, что локаль не была корректно установлена в Главе 6 или не поддерживается Glibc.

```
locale: Cannot set LC_* to default locale: No such file or directory
```

Если Вы видите похожее сообщение, Вам следует или установить необходимую локаль с помощью команды **localedef**, или выбрать другую локаль. Последующие инструкции предполагают, что Вы не получили подобных сообщений об ошибке от Glibc.

Некоторые пакеты, не входящие в круг рассмотрения LFS, могут не поддерживать выбранную локаль. Примером может быть библиотека X (Xlib, часть системы X Window), которая выводит следующее сообщение об ошибке, если локаль не соответствует ни одному из имен символьных карт, определенных в ее внутренних файлах:

```
Warning: locale not supported by Xlib, locale set to C
```

В некоторых случаях Xlib предполагает, что имя символьной карты записало в верхнем регистре и со всеми принятыми дефисами. Например, "UTF-8" вместо "iso88591". Можно попробовать положиться на автоматический подбор подходящей карты символов, удалив имя символьной карты из имени локали (и проверив с помощью **locale charmap**, что это допустимо). Например, Вы можете заменить "de_DE.UTF-85@euro" на "de_DE@euro", чтобы Ваша локаль правильно распознавалась Xlib.

Другие пакеты также могут функционировать неверно (иногда не выводя при этом никаких ошибок), если имя локали не соответствует их ожиданиям. В таких случаях можно извлечь полезную информацию, подсмотрев, как другие дистрибутивы поддерживают Вашу локаль.

После того, как верные настройки локали были определены, создадим файл `/etc/profile`:

```
cat > /etc/profile << "EOF"
# Начало /etc/profile

export LANG=<ll>_<CC>.<charmap><@modifiers>

# Конец /etc/profile
EOF
```

Локали «C» (локаль по умолчанию) и «en_US» (рекомендуемая для пользователей в США) различаются. «C» использует 7-битную кодировку US_ASCII и рассматривает байты с установленным старшим битом как недопустимые символы. Поэтому, например, утилита **ls** в этой локали заменяет их на знаки вопроса. Также при попытке послать письмо, содержащие подобные символы, с помощью Mutt или Pine, будет отправлено сообщение, не соответствующее стандарту RFC (кодировка в письме будет указана как «неизвестная 8-битная»). Поэтому Вы можете использовать локаль «C» только если абсолютно уверены, что Вам никогда не понадобится работать с 8-битными символами.

Локали, основанные на UTF-8, поддерживаются не всеми программами. Мы работаем над документированием, и, по возможности, устранением подобных проблем. Подробнее здесь: *http://www.linuxfromscratch.org/blfs/view/svn/introduction/locale-issues.html*.

7.9. Управление устройствами и модулями в системе LFS

В Chapter 6 мы установили пакет Udev. Перед тем, как углубиться в подробности данного метода управления устройствами, мы вкратце расскажем о более ранних способах, расположив их в историческом порядке.

Ранее системы на базе Linux традиционно использовали метод статически создаваемых устройств, когда огромное количество узлов устройств создавались в директории `/dev` (порой до нескольких тысяч узлов), вне зависимости от реального присутствия всех этих устройств. Это, как правило, совершалось скриптом **MAKEDEV**, который много раз вызывал программу **mknod** с соответствующими мажорными и минорными номерами устройств для каждого устройства, вообще существующего в мире.

При использовании Udev, узлы устройств создаются только для того оборудования, которое было обнаружено ядром. Поскольку эти узлы будут создаваться при каждой загрузке системы, они хранятся в файловой системе `tmpfs` (это виртуальная файловая система, полностью располагающаяся в оперативной памяти). Для узлов устройств не нужно много места, поэтому используемая память крайне мала.

7.9.1. История

В феврале 2000 года новая файловая система `devfs` была включена в ядро 2.3.46 и использовалась на протяжении всей серии стабильных ядер 2.4. Несмотря на свое присутствие в дереве исходников ядра, этот метод динамического создания узлов устройств не получал серьезной поддержки со стороны ключевых разработчиков.

Основными проблемами `devfs` были способы определения, создания и именования устройств. Последняя из них, метод именования узлов устройств, была, пожалуй, самой серьезной. Если уж разрешено настраивать имена, выдаваемые узлам устройств, то политика именования должна быть всецело в ведении системного администратора, а не навязываться ему (администратору) разработчиком. Также файловая система `devfs` серьезно страдает от "гонок", которые возникают из-за особенностей ее дизайна и не могут быть устранены без существенного пересмотра ядра. Она долгое время была помечена как устаревшая – в связи с отсутствием поддержки – и в конце концов была полностью удалена из ядра в июне 2006.

С разработкой нестабильной ветки ядра 2.5, позже влившейся в серию стабильных ядер 2.6, была введена новая виртуальная файловая система sysfs. Задача sysfs заключается в предоставлении процессам пользовательского уровня информации об аппаратном обеспечении компьютера. Благодаря этому возможность замены devfs на систему, работающую в пространстве пользователя, стала гораздо более реальной.

7.9.2. Реализация Udev

7.9.2.1. Sysfs

Файловая система sysfs вкратце была описана выше. Любопытный читатель может поинтересоваться - откуда sysfs знает об установленных в системе устройствах и о том, какие номера должны для них использоваться? Драйвера, которые были целиком включены в ядро при компиляции, регистрируют свои объекты в sysfs, как только они будут обнаружены ядром. Те драйвера, которые были скомпилированы в виде подгружаемых модулей, выполняют эту регистрацию при загрузке соответствующего модуля. Как только файловая система sysfs примонтирована (в каталог /sys), данные, предоставляемые встроенными в ядро драйверами через sysfs, становятся доступны процессам пользовательского уровня, в том числе и **udevd**, который на их основе создает узлы устройств.

7.9.2.2. Загрузочный скрипт Udev

Задача загрузочного скрипта **S10udev** состоит в создании узлов устройств при загрузке Linux. Этот скрипт изменяет обработчик событий uevent, которым по умолчанию является **/sbin/hotplug**. Это делается потому, что ядро уже не нуждается в вызове внешней программы для их обработки. Для этого позже будет запущен **udevd**, который начнет прослушивание netlink-сокета в ожидании событий uevent, посылаемых ядром. После этих действий загрузочный скрипт копирует все статические узлы устройств из /lib/udev/devices в /dev. Это необходимо из-за того, что некоторые устройства, директории и символические ссылки должны быть доступны до того, как вступит в силу динамическое управление узлами устройств - на ранних этапах загрузки системы или при запуске собственно **udevd**. Создание статических узлов в /lib/udev/devices также можно использовать как обходной путь для тех устройств, которые не поддерживаются динамической системой управления устройствами. Затем загрузочный скрипт запускает демон Udev, **udevd**, который будет обрабатывать события uevent. В заключение, скрипт заставляет ядро повторить uevent для всех устройств, которые уже были ранее зарегистрированы, и затем ждет, пока **udevd** не обработает их.

7.9.2.3. Создание узлов устройств

При определении правильных мажорных и минорных номеров устройства, Udev полагается на информацию, предоставляемую sysfs в /sys. Пусть, например, файл /sys/class/tty/vcs/dev содержит строку «7:0». Эта строка будет использована **udevd** для создания узла устройства с мажорным номером *7* и минорным номером *0*. Имена и права доступа для узлов, создаваемых в директории /dev, определяются при помощи правил, указанных в файлах из директории /etc/udev/rules.d/. Они нумеруются тем же образом, что и скрипты из пакета LFS-Bootscripts. Если **udevd** не может найти подходящего правила при создании устройства, он установит для

него значения по умолчанию: права - *660*, владелец - *root:root*. Документацию, описывающую синтаксис правил Udev, можно найти в `/usr/share/doc/udev-166/writing_udev_rules/index.html`

7.9.2.4. Загрузка модулей

Драйверы устройств, скомпилированные в виде модулей, могут содержать шаблоны. Шаблоны можно увидеть в выводе команды **modinfo**, и обычно они соответствуют специфичным для шины идентификаторам устройств, поддерживаемых модулем. Например, драйвер *snd-fm801* поддерживает PCI-устройства с идентификатором производителя 0x1319 и идентификатором устройства 0x0801, и имеет шаблон «pci:v00001319d00000801sv*sd*bc04sc01i*». Для большей части оборудования, драйвер шины предоставляет шаблон драйвера, поддерживающего данное устройство, через `sysfs`. Например, файл `/sys/bus/pci/devices/0000:00:0d.0/modalias` может содержать строку «pci:v00001319d00000801sv00001319sd00001319bc04sc01i00». Правила, используемые Udev по умолчанию, заставляют **udevd** вызвать утилиту **/sbin/modprobe**, передав ей содержимое переменной окружения `MODALIAS` (которое должно быть в точности таким же, как и шаблон в соответствующем файле `modalias` в sysfs), что приводит к загрузке всех модулей, которые подходят под этот шаблон.

Для нашего примера это означает, что помимо модуля *snd-fm801* также загрузится устаревший (и нежелательный) драйвер *forte*, если он будет найден. Чуть ниже приведено несколько способов того, как запретить загрузку нежелательных драйверов.

Также, при необходимости, ядро самостоятельно может загружать модули для поддержки интернационализации, сетевых протоколов и файловых систем.

7.9.2.5. Управление подключаемыми динамически устройствами

Когда Вы подключаете устройство, например, USB (Universal Serial Bus) MP3-плеер, ядро распознает, что было подключено новое оборудование, и генерирует событие uevent. Далее uevent обрабатывается **udevd**, как описано выше.

7.9.3. Проблемы с загрузкой модулей и созданием устройств

Существует несколько проблем, с которыми Вы можете столкнуться при использовании системы динамического создания узлов устройств.

7.9.3.1. Модуль ядра не загружается автоматически

Udev сможет загрузить модуль только в том случае, если этот модуль содержит правильный шаблон и драйвер шины предоставляет его посредством `sysfs`. Если это не так, Вам прийдется организовывать загрузку модуля другими средствами. Для ядра Linux-2.6.37, Udev точно не имеет проблем с правильно написанными драйверами для устройств INPUT, IDE, PCI, USB, SCSI, SERIO и FireWire.

To determine if the device driver you require has the necessary support for Udev, run **modinfo** with the module name as the argument. Now try locating the device directory under `/sys/bus` and check whether there is a `modalias` file there.

If the `modalias` file exists in `sysfs`, the driver supports the device and can talk to it directly, but doesn't have the alias, it is a bug in the driver. Load the driver without the help from Udev and expect the issue to be fixed later.

If there is no `modalias` file in the relevant directory under `/sys/bus`, this means that the kernel developers have not yet added modalias support to this bus type. With Linux-2.6.37, this is the case with ISA busses. Expect this issue to be fixed in later kernel versions.

Udev не предназначен для загрузки драйверов-«оберток», таких, как *snd-pcm-oss* и драйверов, не имеющих отношения к аппаратному обеспечению, например, *loop*.

7.9.3.2. Модуль ядра не загружается автоматически и Udev не хочет его загружать

If the «wrapper» module only enhances the functionality provided by some other module (e.g., *snd-pcm-oss* enhances the functionality of *snd-pcm* by making the sound cards available to OSS applications), configure **modprobe** to load the wrapper after Udev loads the wrapped module. To do this, add an «install» line in any `/etc/modprobe.d/<filename>.conf` file. For example:

```
install snd-pcm /sbin/modprobe -i snd-pcm ; \
    /sbin/modprobe snd-pcm-oss ; true
```

If the module in question is not a wrapper and is useful by itself, configure the **S05modules** bootscript to load this module on system boot. To do this, add the module name to the `/etc/sysconfig/modules` file on a separate line. This works for wrapper modules too, but is suboptimal in that case.

7.9.3.3. Udev загружает нежелательный модуль

Либо не собирайте этот модуль, либо занесите его в черный список - файл `/etc/modprobe.d/blacklist.conf`, так же, как ниже в примере с модулем *forte*:

```
blacklist forte
```

При этом модули, занесенные в черный список, могут быть загружены вручную с помощью команды **modprobe**.

7.9.3.4. Udev неверно создает устройство или символическую ссылку

Такое обычно случается, когда некорректно написанное правило ошибочно подходит не тому устройству, для которого оно писалось. Например, плохо написанное правило может соответствовать как SCSI диску (как и задумывалось), так и контроллеру SCSI (а это уже ошибка) из-за того, что отслеживает лишь производителя устройства. Найдите неверно работающее правило и сделайте его более "узким"; в этом Вам может помочь команда **udevadm info**.

7.9.3.5. Правило Udev не всегда срабатывает

Это может быть одним из проявлений предыдущей проблемы. Если это не так, и ваше правило использует атрибуты `sysfs`, возможно, разгадка таится в проблемах с таймингами ядра, и наверняка это будет исправлено в ближайшем будущем. А пока Вы можете обойти это, создав правило, которое будет дожидаться появления необходимого атрибута в `sysfs`, и добавив его в файл `/etc/udev/rules.d/10-wait_for_sysfs.rules` (создайте этот файл, если он еще не существует). Если Вы столкнулись с подобной проблемой и это вам помогло, пожалуйста, сообщите об этом в список рассылки LFS Development.

7.9.3.6. Udev не создает устройство

Дальнейший текст предполагает, что драйвер либо встроен в ядро, либо уже загружен, и что Вы уже проверили, что Udev не создает устройства с неверным именем.

Udev не имеет информации, необходимой для создания узла, если драйвер ядра не предоставляет данные об устройстве через `sysfs`. Чаще всего такое случается со сторонними драйверами, не входящими в дерево исходников ядра. В подобной ситуации Вам может помочь создание статического узла в директории `/lib/udev/devices` с правильными мажорными/минорными номерами (обратитесь к файлу `devices.txt` из документации ядра, или к информации, предоставляемой поставщиком драйвера). Он, как и все статические узлы, будет скопирован в директорию `/dev` загрузочным скриптом **S10udev**.

7.9.3.7. Порядок именования устройств случайным образом меняется после перезагрузки

Это происходит потому, что Udev, в силу своего дизайна, обрабатывает события uevent и загружает модули параллельно, в непредсказуемом порядке. Это никогда не будет «исправлено». Вам не следует полагаться на то, что имена устройств всегда будут оставаться одними и теми же. Вместо этого лучше напишите свои собственные правила, которые будут создавать стабильные символические ссылки основываясь на некоторых постоянных атрибутах устройств, например, серийных номерах или выводе различных *_id утилит, установленных вместе с Udev. За примерами обратитесь к Раздел 7.10, «Создание собственных ссылок на устройства» и Раздел 7.13, «Configuring the network Script».

7.9.4. Полезно прочитать

Дополнительная информация доступна на следующих сайтах:

- Реализация `devfs` на пользовательском уровне *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*
- Перечень дополнительных материалов *http://www.kernel.org/pub/linux/utils/kernel/hotplug/udev.html*

7.10. Создание собственных ссылок на устройства

7.10.1. Ссылки на CD-ROM

Некоторые программы, которые Вы можете захотеть установить позже (например, разные медиаплееры), ожидают, что ссылки `/dev/cdrom` и `/dev/dvd` существуют и указывают на устройство CD-ROM или DVD-ROM. Кроме того, эти ссылки удобно использовать в файле `/etc/fstab`. Вместе с Udev поставляется скрипт, который сгенерирует файлы правил для создания этих символических ссылок основываясь на возможностях каждого устройства, однако Вам необходимо выбрать один из двух режимов работы скрипта.

Во-первых, скрипт может работать в режиме «by-path» (используется большинством устройств USB и FireWire), когда создаваемые им правила будут зависеть от физического пути к CD- или DVD-устройству. Во-вторых, он может работать в режиме «by-id» (применяется для большинства устройств IDE и SCSI), при котором правила будут зависеть от идентификатора, сохраненного в самом устройстве. Путь определяется скриптом **path_id** из поставки Udev, и идентификатор считывается при помощи его же программ **ata_id** или **scsi_id**, в зависимости от типа Вашего устройства.

У каждого подхода есть свои преимущества; выбор будет зависеть от того, какого типа изменения будут происходить с Вашим устройством наиболее часто. Если физический путь к устройству может смениться (например, потому что Вы подключаете его в разные порты/слоты), то Вам следует использовать режим «by-id». С другой стороны, если измениться может идентификатор устройства, например, Вы ожидаете его скорой поломки и планируете заменить его другим таким же, подключив в тот же самый разъем, то Вам подойдет режим «by-path».

Если возможны изменения и того, и другого характера, Вам прийдется выбирать режим, основываясь на предположениях о том, какие же все-таки будут происходить чаще.

Важно

External devices (for example, a USB-connected CD drive) should not use by-path persistence, because each time the device is plugged into a new external port, its physical path will change. All externally-connected devices will have this problem if you write Udev rules to recognize them by their physical path; the problem is not limited to CD and DVD drives.

If you wish to see the values that the Udev scripts will use, then for the appropriate CD-ROM device, find the corresponding directory under `/sys` (e.g., this can be `/sys/block/hdd`) and run a command similar to the following:

```
udevadm test /sys/block/hdd
```

Look at the lines containing the output of various *_id programs. The «by-id» mode will use the ID_SERIAL value if it exists and is not empty, otherwise it will use a combination of ID_MODEL and ID_REVISION. The «by-path» mode will use the ID_PATH value.

If the default mode is not suitable for your situation, then the following modification can be made to the `/lib/udev/rules.d/75-cd-aliases-generator.rules` file, as follows (where *mode* is one of «by-id» or «by-path»):

```
sed -i -e 's/"write_cd_rules"/"write_cd_rules mode"/' \
    /lib/udev/rules.d/75-cd-aliases-generator.rules
```

Note that it is not necessary to create the rules files or symlinks at this time, because you have bind-mounted the host's `/dev` directory into the LFS system, and we assume the symlinks exist on the host. The rules and symlinks will be created the first time you boot your LFS system.

However, if you have multiple CD-ROM devices, then the symlinks generated at that time may point to different devices than they point to on your host, because devices are not discovered in a predictable order. The assignments created when you first boot

the LFS system will be stable, so this is only an issue if you need the symlinks on both systems to point to the same device. If you need that, then inspect (and possibly edit) the generated /etc/udev/rules.d/70-persistent-cd.rules file after booting, to make sure the assigned symlinks match what you need.

7.10.2. Dealing with duplicate devices

As explained in Раздел 7.9, «Управление устройствами и модулями в системе LFS», the order in which devices with the same function appear in /dev is essentially random. E.g., if you have a USB web camera and a TV tuner, sometimes /dev/video0 refers to the camera and /dev/video1 refers to the tuner, and sometimes after a reboot the order changes to the opposite one. For all classes of hardware except sound cards and network cards, this is fixable by creating udev rules for custom persistent symlinks. The case of network cards is covered separately in Раздел 7.13, «Configuring the network Script», and sound card configuration can be found in *BLFS*.

For each of your devices that is likely to have this problem (even if the problem doesn't exist in your current Linux distribution), find the corresponding directory under /sys/class or /sys/block. For video devices, this may be /sys/class/video4linux/videoX. Figure out the attributes that identify the device uniquely (usually, vendor and product IDs and/or serial numbers work):

```
udevadm info -a -p /sys/class/video4linux/video0
```

Then write rules that create the symlinks, e.g.:

```
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
```

The result is that /dev/video0 and /dev/video1 devices still refer randomly to the tuner and the web camera (and thus should never be used directly), but there are symlinks /dev/tvtuner and /dev/webcam that always point to the correct device.

7.11. Настройка скрипта localnet

Часть работы скрипта **localnet** заключается в установке имени системы, которое определяется в файле /etc/sysconfig/network.

Создадим файл /etc/sysconfig/network и зададим имя системы командой:

```
echo "HOSTNAME=<lfs>" > /etc/sysconfig/network
```

<lfs> нужно заменить на имя, которое Вы хотите дать компьютеру. Не вводите здесь полное доменное имя (Fully Qualified Domain Name, FQDN). Эту информацию мы поместим в файл /etc/hosts в следующей секции.

7.12. Создание файла /etc/hosts

Если Вы собираетесь настраивать сетевую карту, Вам необходимо решить, какие IP-адрес, полное доменное имя (FQDN) и возможные псевдонимы для него включить в файл /etc/hosts. Синтаксис файла такой:

```
IP_адрес myhost.example.org псевдонимы
```

Если компьютер не будет видим в Интернете (например, вдруг Вы имеете зарегистрированный домен и доступный выделенный блок IP-адресов—у большинства пользователей этого нет), убедитесь, что IP-адрес находится в диапазоне, выделенном для частных сетей. Верными диапазонами являются:

```
Частная сеть Диапазон адресов      Префикс
10.0.0.1 - 10.255.255.254           8
172.x.0.1 - 172.x.255.254           16
192.168.y.1 - 192.168.y.254         24
```

x может быть любым числом из диапазона 16-31. y может быть любым числом из диапазона 0-255.

Верным частным IP-адресом может быть 192.168.1.1. Верным FQDN для этого IP может быть lfs.example.org.

Даже если Вы не имеете сетевой карты, все равно необходимо указать FQDN. Это необходимо для корректной работы некоторых программ.

Создайте файл /etc/hosts командой:

```
cat > /etc/hosts << "EOF"
# Начало /etc/hosts (версия для сетевой карты)

127.0.0.1 localhost
<192.168.1.1> <HOSTNAME.example.org> [hostname1] [hostname2 ...]

# Конец /etc/hosts (версия для сетевой карты)
EOF
```

Необходимо заменить значения *<192.168.1.1>* и *<HOSTNAME.example.org>* на соответствующие Вашим условиям (если IP-адрес был присвоен сетевым/системным администратором и машина будет подключена к существующей сети). Необязательные имена псевдонимов могут быть опущены.

Если Вы не собираетесь настраивать сетевую карту или у Вас ее нет, создайте файл /etc/hosts командой:

```
cat > /etc/hosts << "EOF"
# Начало /etc/hosts (версия без сетевой карты)

127.0.0.1 <HOSTNAME.example.org> <HOSTNAME> localhost

# Конец /etc/hosts (версия без сетевой карты)
EOF
```

7.13. Configuring the network Script

This section only applies if a network card is to be configured.

If a network card will not be used, there is likely no need to create any configuration files relating to network cards. If that is the case, remove the `network` symlinks from all run-level directories (`/etc/rc.d/rc*.d`).

7.13.1. Creating stable names for network interfaces

With Udev and modular network drivers, the network interface numbering is not persistent across reboots by default, because the drivers are loaded in parallel and, thus, in random order. For example, on a computer having two network cards made by Intel and Realtek, the network card manufactured by Intel may become `eth0` and the Realtek card becomes `eth1`. In some cases, after a reboot the cards get renumbered the other way around. To avoid this, Udev comes with a script and some rules to assign stable names to network cards based on their MAC address.

Pre-generate the rules to ensure the same names get assigned to the same devices at every boot, including the first:

```
for NIC in /sys/class/net/* ; do
    INTERFACE=${NIC##*/} udevadm test --action=add $NIC
done
```

Now, inspect the `/etc/udev/rules.d/70-persistent-net.rules` file, to find out which name was assigned to which network device:

```
cat /etc/udev/rules.d/70-persistent-net.rules
```

The file begins with a comment block followed by two lines for each NIC. The first line for each NIC is a commented description showing its hardware IDs (e.g. its PCI vendor and device IDs, if it's a PCI card), along with its driver in parentheses, if the driver can be found. Neither the hardware ID nor the driver is used to determine which name to give an interface; this information is only for reference. The second line is the Udev rule that matches this NIC and actually assigns it a name.

All Udev rules are made up of several keys, separated by commas and optional whitespace. This rule's keys and an explanation of each of them are as follows:

- `SUBSYSTEM=="net"` - This tells Udev to ignore devices that are not network cards.
- `ACTION=="add"` - This tells Udev to ignore this rule for a uevent that isn't an add ("remove" and "change" uevents also happen, but don't need to rename network interfaces).
- `DRIVERS=="?*"` - This exists so that Udev will ignore VLAN or bridge sub-interfaces (because these sub-interfaces do not have drivers). These sub-interfaces are skipped because the name that would be assigned would collide with their parent devices.
- `ATTR{address}` - The value of this key is the NIC's MAC address.
- `ATTR{type}=="1"` - This ensures the rule only matches the primary interface in the case of certain wireless drivers, which create multiple virtual interfaces. The secondary interfaces are skipped for the same reason that VLAN and bridge sub-interfaces are skipped: there would be a name collision otherwise.
- `KERNEL=="eth*"` - This key was added to the Udev rule generator to handle machines that have multiple network interfaces, all with the same MAC address (the PS3 is one such machine). If the independent interfaces have different basenames, this key will allow Udev to tell them apart. This is generally not necessary for most Linux From Scratch users, but does not hurt.

• NAME - The value of this key is the name that Udev will assign to this interface.

The value of NAME is the important part. Make sure you know which name has been assigned to each of your network cards before proceeding, and be sure to use that NAME value when creating your configuration files below.

7.13.2. Creating Network Interface Configuration Files

Which interfaces are brought up and down by the network script depends on the files and directories in the /etc/sysconfig/network-devices hierarchy. This directory should contain a sub-directory for each interface to be configured, such as ifconfig.xyz, where «xyz» is a network interface name. Inside this directory would be files defining the attributes to this interface, such as its IP address(es), subnet masks, and so forth.

The following command creates a sample ipv4 file for the *eth0* device:

```
cd /etc/sysconfig/network-devices
mkdir -v ifconfig.eth0
cat > ifconfig.eth0/ipv4 << "EOF"
ONBOOT=yes
SERVICE=ipv4-static
IP=192.168.1.1
GATEWAY=192.168.1.2
PREFIX=24
BROADCAST=192.168.1.255
EOF
```

The values of these variables must be changed in every file to match the proper setup. If the ONBOOT variable is set to «yes» the network script will bring up the Network Interface Card (NIC) during booting of the system. If set to anything but «yes» the NIC will be ignored by the network script and not be brought up.

The SERVICE variable defines the method used for obtaining the IP address. The LFS-Bootscripts package has a modular IP assignment format, and creating additional files in the /etc/sysconfig/network-devices/services directory allows other IP assignment methods. This is commonly used for Dynamic Host Configuration Protocol (DHCP), which is addressed in the BLFS book.

The GATEWAY variable should contain the default gateway IP address, if one is present. If not, then comment out the variable entirely.

The PREFIX variable needs to contain the number of bits used in the subnet. Each octet in an IP address is 8 bits. If the subnet's netmask is 255.255.255.0, then it is using the first three octets (24 bits) to specify the network number. If the netmask is 255.255.255.240, it would be using the first 28 bits. Prefixes longer than 24 bits are commonly used by DSL and cable-based Internet Service Providers (ISPs). In this example (PREFIX=24), the netmask is 255.255.255.0. Adjust the PREFIX variable according to your specific subnet.

7.13.3. Creating the /etc/resolv.conf File

If the system is going to be connected to the Internet, it will need some means of Domain Name Service (DNS) name resolution to resolve Internet domain names to IP addresses, and vice versa. This is best achieved by placing the IP address of the DNS server, available from the ISP or network administrator, into `/etc/resolv.conf`. Create the file by running the following:

```
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
```

Replace *<IP address of the nameserver>* with the IP address of the DNS most appropriate for the setup. There will often be more than one entry (requirements demand secondary servers for fallback capability). If you only need or want one DNS server, remove the second *nameserver* line from the file. The IP address may also be a router on the local network.

Глава 8. Делаем LFS-систему загружаемой

8.1. Вступление

Пришло время сделать LFS-систему способной к загрузке. Эта глава описывает создание файла fstab, сборку ядра для новой LFS-системы и установку загрузчика GRUB, который позволит Вам выбирать для запуска Вашу LFS-систему при включении компьютера.

8.2. Создание файла /etc/fstab

Файл /etc/fstab используется некоторыми программами для определения того, куда по умолчанию должны быть смонтированы файловые системы, в каком порядке и должна ли производиться проверка (на целостность) перед монтированием. Создадим новую таблицу файловых систем:

```
cat > /etc/fstab << "EOF"
# Начало /etc/fstab

# файловая система  точка монтирования  тип    опции          дамп  порядок
#                                                                   проверки

/dev/<xxx>          /                    <fff>  defaults        1     1
/dev/<yyy>          swap                 swap   pri=1           0     0
proc                /proc                proc   defaults        0     0
sysfs               /sys                 sysfs  defaults        0     0
devpts              /dev/pts             devpts gid=4,mode=620  0     0
tmpfs               /dev/shm             tmpfs  defaults        0     0
# Конец /etc/fstab
EOF
```

Замените *<xxx>*, *<yyy>* и *<fff>* значениями, верными для вашей системы, например, hda2, hda5 и ext3. За более подробным описанием шестой колонки в этом файле обратитесь к **man 5 fstab**.

Точка монтирования /dev/shm для tmpfs добавлена для того, чтобы позволить приложениям использовать POSIX-разделяемую память. Ядро должно быть собрано с поддержкой данной возможности (подробнее об этом - в следующей секции). На самом деле, очень небольшое число приложений используют POSIX-разделяемую память, поэтому подключение tmpfs на /dev/shm не является обязательным. Поробнее об этом Вы можете узнать из файла Documentation/filesystems/tmpfs.txt, находящемся в дереве исходников ядра.

Чтобы имена файлов, содержащие не-ASCII символы, отображались верно, для файловых систем, ведущих свою историю от ОС MS-DOS или Windows (например, vfat, ntfs, smbfs, cifs, iso9660, udf) необходимо указать опцию монтирования «iocharset». Значение этой опции должно совпадать с названием кодировки, применяемой в вашей локали (возможно, его прийдется немного изменить - названия кодировок локалей и кодировок ядра не всегда совпадают). Это возможно, если соответствующее описание кодировки было включено в ядро на этапе конфигурации (File systems -> Native Language Support). Также для файловых систем vfat и smbfs необходима опция «codepage». Она должна указывать кодировку,

используемую в системе MS-DOS в Вашей стране. Например, для корректного монтирования USB Flash дисков, пользователю с локалью ru_RU.KOI8-R следует добавить следующие опции в файл `/etc/fstab` в строки, соответствующие этим дискам:

```
noauto,user,quiet,showexec,iocharset=koi8r,codepage=866
```

Пользователю с локалью ru_RU.UTF-8 - такие:

```
noauto,user,quiet,showexec,iocharset=utf8,codepage=866
```

Замечание

Во втором случае ядро выведет следующее сообщение:

```
FAT: utf8 is not a recommended IO charset for FAT filesystems,
    filesystem will be case sensitive!
```

Его следует проигнорировать, поскольку любые другие значения «iocharset» приведут к неверному отображению имен файлов в локалях, основанных на UTF-8.

Также Вы можете указать кодировку по умолчанию и значения iocharset для некоторых файловых систем во время конфигурирования ядра. Ответственные за это параметры называются «Default NLS Option» (`CONFIG_NLS_DEFAULT`), «Default Remote NLS Option» (`CONFIG_SMB_NLS_DEFAULT`), «Default codepage for FAT» (`CONFIG_FAT_DEFAULT_CODEPAGE`) и «Default iocharset for FAT» (`CONFIG_FAT_DEFAULT_IOCHARSET`). Для файловой системы NTFS невозможно таким образом указать эти значения.

Для некоторых типов жестких дисков имеется возможность сделать файловую систему ext3 более устойчивой к сбоям питания. Для этого добавьте параметр `barrier=1` к соответствующей записи в `/etc/fstab`. Чтобы проверить, поддерживает ли жесткий диск такую опцию, запустите утилиту *hdparm*, передав ей параметром его имя. Например, если вывод

```
hdparm -I /dev/sda | grep NCQ
```

не пуст, то жесткий диск поддерживает данную возможность.

Замечание: разделы, располагающиеся на Logical Volume Management (LVM), не могут использовать параметр `barrier`.

8.3. Linux-2.6.37

Пакет Linux содержит ядро Linux.

Приблизительное время сборки:	1.0 - 5.0 SBU
Требует свободного места на диске:	540 - 800 MB

8.3.1. Установка ядра

Сборка ядра состоит из нескольких шагов—конфигурирования, компиляции и установки. Обратитесь к файлу README, который находится в дереве исходников ядра, если Вас интересуют альтернативные методы конфигурирования ядра, не описываемые в этой книге.

Подготовим ядро к компиляции следующей командой:

```
make mrproper
```

Эта команда очищает дерево исходных кодов ядра от возможно присутствующих в нем временных файлов. Команда разработчиков ядра рекомендует выполнять это действие перед каждой компиляцией. Не стоит полагаться на то, что дерево исходников будет чисто после распаковки архива - это не всегда так.

Конфигурирование ядра будет выполняться при помощи псевдографического интерфейса. Основная информация о параметрах конфигурации и способах ее выполнения изложена здесь: *http://www.linuxfromscratch.org/hints/downloads/files/kernel-configuration.txt*. Также в книге BLFS есть некоторая информация о требованиях пакетов, не входящих в LFS, к опциям ядра - *http://www.linuxfromscratch.org/blfs/view/svn/longindex.html#kernel-config-index*:

```
make LANG=<host_LANG_value> LC_ALL= menuconfig
```

Значения параметров make:

LANG=<host_LANG_value> LC_ALL=
: Устанавливает ту же локаль для использования, что и на хост-системе. Это необходимо для правильного отображения линий интерфейса ncurses в menuconfig для локалей, основанных на UTF-8.
  Не забудьте заменить *<host_LANG_value>* значением переменной $LANG из окружения Вашей хост-системы. Если эта переменная не установлена, вместо нее можно использовать значение переменной $LC_ALL или $LC_CTYPE.

В некоторых ситуациях **make oldconfig** может быть более подходящим вариантом. Обратитесь к файлу README за более поробными разъяснениями.

Вы можете пропустить конфигурирование ядра, скопировав файл настройки .config (если, конечно, он у Вас имеется) из хост-системы в директорию исходников linux-2.6.37. Однако, мы не рекомендуем этот вариант. Гораздо лучше вручную пройтись по всем настройкам ядра и создать конфигурацию с нуля.

Скомпилируем ядро и модули:

```
make
```

Если Вы используете модули ядра, возможно, Вам понадобится дополнительно настроить их в `/etc/modprobe.d`. Информацию о настройке модулей и самого ядра Вы можете подчерпнуть из Раздел 7.9, «Управление устройствами и модулями в системе LFS» и документации ядра в директории `linux-2.6.37/Documentation`. Также, возможно, Вас заинтересует страница `modprobe.conf(5)`.

Установим модули, если конфигурация ядра подразумевает их использование:

```
make modules_install
```

После того, как компиляция ядра завершена, необходимо выполнить несколько дополнительных шагов для завершения установки. Нужно скопировать некоторые файлы в директорию `/boot`.

Путь к образу ядра на разных платформах может сильно отличаться. Вы можете свободно изменить имя файла, используемое ниже, но оно обязательно должно начинаться со слова *vmlinux*, чтобы не возникло проблем с автоматической настройкой загрузчика в следующей секции. Следующая команда предполагает, что Вы используете архитектуру x86:

```
cp -v arch/x86/boot/bzImage /boot/vmlinux-2.6.37-lfs-6.8
```

Файл `System.map` содержит карту символов ядра. В нем указаны точки входа для всех функций API и адреса структур данных для запущенного ядра. Он полезен при исследовании возникающих в системе неполадок. Выполните команду ниже для установки map-файла:

```
cp -v System.map /boot/System.map-2.6.37
```

Файл настроек `.config`, созданный выше на шаге **make menuconfig**, содержит значения всех опций конфигурации для только что собранного ядра. Неплохой идеей будет сохранить его на будущее:

```
cp -v .config /boot/config-2.6.37
```

Установим документацию для ядра Linux:

```
install -d /usr/share/doc/linux-2.6.37
cp -r Documentation/* /usr/share/doc/linux-2.6.37
```

Важно помнить, что файлы в директории исходных кодов ядра не принадлежат пользователю *root*. Даже если архив был распакован из-под пользователя *root* (как мы и сделали в chroot-окружении), файлы будут иметь те же идентификаторы пользователя и группы, что и на компьютере того, кто упаковывал этот архив. Для любого другого пакета это не является проблемой, поскольку дерево исходников удаляется сразу после установки. Но, как правило, исходные коды ядра Linux остаются в системе надолго. Поэтому существует вероятность, что UID, который имеют файлы ядра, будет присвоен какому-либо пользователю на машине. Этот пользователь будет иметь доступ на запись в исходные файлы ядра.

Если Вы собираетесь оставить дерево исходников ядра, выполните команду **chown -R 0:0** над директорией `linux-2.6.37`, чтобы быть уверенным, что все файлы доступны только пользователю *root*.

Внимание

В некоторых разделах документации Вы можете наткнуться на рекомендацию создать символическую ссылку `/usr/src/linux`, указывающую на каталог с исходниками ядра. Эта ссылка специфична для более ранних ядер, чем 2.6, и *не должна* присутствовать в системе LFS, поскольку из-за нее может возникнуть проблема со сборкой некоторых пакетов, которые, возможно, понадобятся Вам по окончании сборки LFS.

Внимание

Файлы в системной директории `include` должны *всегда* быть именно теми подготовленными заголовочными файлами из архива ядра Linux, которые использовались при сборке Glibc. *Ни в коем случае* нельзя заменять их "сырыми" заголовочными файлами или заголовочными файлами другой версии ядра!

8.3.2. Настройка очередности загрузки модулей Linux

Если USB драйверы (ehci_hcd, ohci_hcd и uhci_hcd) были собраны в виде модулей, необходимо создать файл `/etc/modprobe.d/usb.conf` для того, чтобы обеспечить их загрузку в правильном порядке. Модуль ehci_hcd должен быть загружен перед ohci_hcd и uhci_hcd во избежание появления предупреждения при загрузке системы.

Создадим новый файл `/etc/modprobe.d/usb.conf` следующими командами:

```
install -v -m755 -d /etc/modprobe.d
cat > /etc/modprobe.d/usb.conf << "EOF"
# Начало /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

# Конец /etc/modprobe.d/usb.conf
EOF
```

8.3.3. Содержимое пакета Linux

Установленные файлы: config-2.6.37, vmlinux-2.6.37-lfs-6.8-2.6.37 и System.map-2.6.37

Установленные каталоги: /lib/modules, /usr/share/doc/linux-2.6.37

Краткое описание

`config-2.6.37`	Содержит всю конфигурацию ядра
`vmlinux-2.6.37-lfs-6.8`	Ядро системы Linux. Оно является самой первой частью операционной системы, загружаемой в оперативную память после включения компьютера. Ядро определяет и настраивает все компоненты аппаратного обеспечения компьютера, а затем делает их доступными для программ в виде дерева файлов, превращая процессор

	в многозадачную машину, способную выполнять множество программ "одновременно" с точки зрения пользователя
System.map-2.6.37	Список адресов и символов; из него можно узнать точки входа и адреса всех функций и структур данных ядра.

8.4. Using GRUB to Set Up the Boot Process

8.4.1. Introduction

Boot loading can be a complex area, so a few cautionary words are in order. Be familiar with the current boot loader and any other operating systems present on the hard drive(s) that need to be bootable. Make sure that an emergency boot disk is ready to «rescue» the computer if the computer becomes unusable (un-bootable).

The procedure involves writing some special GRUB files to specific locations on the hard drive. We highly recommend creating a GRUB boot floppy diskette as a backup. Insert a blank floppy diskette and run the following commands:

```
cd /tmp
grub-mkrescue --output=grub-img.iso
dd if=grub-img.iso of=/dev/fd0 bs=1440 count=1
```

Alternatively, a boot CD can be created by using your host system's CD burning tools to burn the `grub-img.iso` on to a blank CD.

GRUB uses its own naming structure for drives and partitions in the form of *(hdn,m)*, where *n* is the hard drive number and *m* is the partition number. The hard drive number starts from zero, but the partition number starts from one for normal partitions and five for extended partitions. Note that this is different from earlier versions where both numbers started from zero. For example, partition `sda1` is *(hd0,1)* to GRUB and `sdb3` is *(hd1,3)*. In contrast to Linux, GRUB does not consider CD-ROM drives to be hard drives. For example, if using a CD on `hdb` and a second hard drive on `hdc`, that second hard drive would still be *(hd1)*.

You can determine what GRUB thinks your disk devices are by running:

```
grub-mkdevicemap --device-map=device.map
cat device.map
```

The location of the boot partition is a choice of the user that affects the configuration. One recommendation is to have a separate small (suggested size is 100 MB) partition just for boot information. That way each build, whether LFS or some commercial distro, can access the same boot files and access can be made from any booted system. If you choose to do this, you will need to mount the separate partition, move all files in the current `/boot` directory (e.g. the linux kernel you just built in the previous section) to the new partition. You will then need to unmount the partition and remount it as `/boot`. If you do this, be sure to update `/etc/fstab`.

Using the current lfs partition will also work, but configuration for multiple systems is more difficult.

8.4.2. Setting Up the Configuration

Using the above information, determine the appropriate designator for the root partition (or boot partition, if a separate one is used). For the following example, it is assumed that the root (or separate boot) partition is `sda2`.

Install the GRUB files into `/boot/grub`:

```
grub-install --grub-setup=/bin/true /dev/sda
```

We use --grub-setup=/bin/true for now to prevent updating the Master Boot Record (MBR). In this way, we can test our installation before committing to a change that is hard to revert.

Generate `/boot/grub/grub.cfg`:

```
grub-mkconfig -o /boot/grub/grub.cfg
```

Here **grub-mkconfig** uses the files in `/etc/grub.d/` to determine the contents of this file. The configuration file will look something like:

```
#
# DO NOT EDIT THIS FILE
#
# It is automatically generated by /usr/sbin/grub-mkconfig using templates
# from /etc/grub.d and settings from /etc/default/grub
#

### BEGIN /etc/grub.d/00_header ###
set default=0
set timeout=5
### END /etc/grub.d/00_header ###

### BEGIN /etc/grub.d/10_linux ###
menuentry "GNU/Linux, Linux 2.6.37-lfs-6.8" {
        insmod ext2
        set root=(hd0,2)
        search --no-floppy --fs-uuid --set 915852a7-859e-45a6-9ff0-d3ebfdb5cea2
        linux   /boot/vmlinux-2.6.37-lfs-6.8 root=/dev/sda2 ro
}
menuentry "GNU/Linux, Linux 2.6.37-lfs-6.8" (recovery mode)" {
        insmod ext2
        set root=(hd0,2)
        search --no-floppy --fs-uuid --set 915852a7-859e-45a6-9ff0-d3ebfdb5cea2
        linux   /boot/vmlinux-2.6.37-lfs-6.8 root=/dev/sda2 ro single
}
menuentry "GNU/Linux, Linux 2.6.28-11-server" {
        insmod ext2
        set root=(hd0,2)
        search --no-floppy --fs-uuid --set 6b4c0339-5501-4a85-8351-e398e5252be8
        linux   /boot/vmlinuz-2.6.28-11-server root=UUID=6b4c0339-5501-4a85-8351-e398e5252be8 ro
        initrd  /boot/initrd.img-2.6.28-11-server
}
menuentry "GNU/Linux, Linux 2.6.28-11-server (recovery mode)" {
        insmod ext2
        set root=(hd0,2)
        search --no-floppy --fs-uuid --set 6b4c0339-5501-4a85-8351-e398e5252be8
        linux   /boot/vmlinuz-2.6.28-11-server root=UUID=6b4c0339-5501-4a85-8351-e398e5252be8 ro single
        initrd  /boot/initrd.img-2.6.28-11-server
}
### END /etc/grub.d/10_linux ###

### BEGIN /etc/grub.d/30_os-prober ###
### END /etc/grub.d/30_os-prober ###

### BEGIN /etc/grub.d/40_custom ###
# This file provides an easy way to add custom menu entries.  Simply type the
# menu entries you want to add after this comment.  Be careful not to change
# the 'exec tail' line above.
### END /etc/grub.d/40_custom ###
```

Замечание

- Even though there is a warning not to edit the file, you can do so as long as you do not re-run **grub-mkconfig**.
- The *search* lines are generally not useful for LFS systems as that command only sets an internal GRUB variable used to find the kernel image. The *set root* command provides the same capability without the overhead of searching.
- The *set root* and *insmod ext2* commands can be moved out of the *menuentry* sections to apply to all sections of the file. This leads to a simple section like:

```
menuentry "Linux 2.6.37-lfs-6.8" {
linux   /boot/vmlinux-2.6.37-lfs-6.8 root=/dev/sda2 ro
}
```

- Passing a UUID to the kernel requires an initial ram disk (initrd) not built by LFS.
- If the `/boot` partition is installed on a separate partition, the linux and initrd lines should not have the string */boot* prefixed to the file names.
- In this example the kernel files for a Ubuntu installation are also found in `/boot`.

8.4.3. Testing the Configuration

The core image of GRUB is also a Multiboot kernel, so if you already have *GRUB Legacy* loaded you can load GRUB-1.98 through your old boot loader. To accomplish this, you will need to exit the **chroot** environment now and re-enter it in the next section to finish the few remaining portions of the book.

```
/sbin/reboot
...
grub> root (hd0,1)
grub> kernel /boot/grub/core.img
grub> boot
```

Note that the GRUB commands above are assumed to be GRUB Legacy. At this point the GRUB prompt will appear (very similar to GRUB Legacy) and you can explore the interface or boot to one of the systems in the grub.cfg file.

8.4.4. Updating the Master Boot Record

If you tested the GRUB configuration as specified above, re-enter the **chroot** environment.

Внимание

The following command will overwrite the current boot loader. Do not run the command if this is not desired, for example, if using a third party boot manager to manage the Master Boot Record (MBR).

Update the MBR with:

```
grub-setup '<DEVICE>'
```

Change the DEVICE above to your boot disk, normally '(hd0)' or /dev/sda. If using (hd0) be sure to escape the parentheses with backslashes or single quotes to prevent the shell from interpreting them as a sub-shell.

This program uses the following defaults and are correct if you did not deviate from the instructions above:

- boot image - boot.img
- core image - core.img
- directory - /boot/grub
- device map - device.map
- default root setting - guessed

Замечание

The root setting is the default value if a 'set root' instruction is not found in grub.cfg. This is the partition that is searched for the kernel and other supporting files. It is different from the 'root=' parameter on the 'linux' line in the configuration line. The latter is the partition the kernel mounts as '/'. In the example grub.cfg above, both values point to /dev/sda2, but if there is a separate boot partition, they will be different.

Глава 9. Конец

9.1. Конец

Хорошо сработано! Новая LFS-система установлена! Мы хотим пожелать Вам удачи с Вашей новой блестящей самосборной Linux-системой.

Хорошей идеей будет создать файл `/etc/lfs-release`. Имея этот файл, Вам (и нам тоже, если Вам понадобится помощь), будет легко определить, какая версия LFS установлена. Создайте этот файл следующей командой:

```
echo 6.8 > /etc/lfs-release
```

9.2. Регистрация

Теперь, когда Вы закончили сборку собственной системы, не хотели бы Вы зарегистрироваться как пользователь LFS? Зайдите на *http://www.linuxfromscratch.org/cgi-bin/lfscounter.cgi* и зарегистрируйтесь как пользователь LFS, введя Ваше имя и первую версию LFS, которую Вы использовали.

Пришло время перезагрузиться в LFS.

9.3. Перезагрузка системы

Теперь, когда все программное обеспечение установлено, пришло время перезагрузить Ваш компьютер. Однако, помните несколько вещей. Система, которую Вы создали по этой книге, минимальна, и скорее всего не имеет необходимых возможностей для продолжения движения вперед. Установив несколько дополнительных пакетов из книги BLFS, находясь пока в нашем временном окружении, Вы можете оказаться в лучшем положении, когда перезагруизтесь в Вашу свежеустановленную LFS-систему. Установив текстовый веб-браузер, например Lynx, Вы сможете просматривать книгу BLFS в одном виртуальном терминале, собирая пакеты в другом. Пакет GPM позволит вам переносить текст путем копирования/вставки между виртуальными терминалами. Наконец, если Вы не имеете статического IP, установка Dhcpcd и PPP также будет весьма кстати.

После всего вышесказанного, начнем продвигаться к первой загрузке в нашу блестящую установленную LFS! Сначала покинем временное окружение:

```
logout
```

Затем отмонтируем виртуальные файловые системы:

```
umount -v $LFS/dev/pts
umount -v $LFS/dev/shm
umount -v $LFS/dev
umount -v $LFS/proc
umount -v $LFS/sys
```

Отмонтируем саму файловую систему LFS:

```
umount -v $LFS
```

Если было создано несколько разделов, отключите их перед отмонтированием основного, примерно так:

```
umount -v $LFS/usr
umount -v $LFS/home
umount -v $LFS
```

Наконец, перезагрузите систему:

```
shutdown -r now
```

Если загрузчик GRUB был установлен в соответствии с инструкциями выше, в меню автоматически будет выбрана опция загрузки *LFS 6.8*.

Когда перезагрузка завершится, LFS-система будет готова к использованию и установке дополнительного программного обеспечения.

9.4. Что дальше?

Спасибо Вам за прочтение книги LFS. Мы надеемся, что Вы нашли эту книгу полезной и узнали много нового о процессе построения системы Linux.

Теперь, когда LFS-система установлена, Вы можете задаться вопросом «А что же дальше?» Чтобы ответить на этот вопрос, мы собрали для Вас список ресурсов.

- Поддержка

  Отчеты об ошибках и угрозах безопасности регулярно появляются для любого программного обеспечения. Поскольку LFS-система компилируется из исходников, Вы можете не отставать от этих сообщений. Существуют несколько онлайн-ресурсов, которые следят за такими отчетами, ссылки на некоторые из них приведены ниже:

  - Freshmeat.net (*http://freshmeat.net/*)

    Freshmeat может сообщать Вам (по E-mail) о выходе новых версий пакетов, установленных в Вашей системе.

  - *CERT* (Computer Emergency Response Team)

    CERT имеет список рассылки, в котором публикуются предупреждения об уязвимостях в различных операционных системах и приложениях. Информация о подписке доступна на *http://www.us-cert.gov/cas/signup.html*.

  - Bugtraq

    Bugtraq - полностью открытый список рассылки, посвященный вопросам компьютерной безопасности. Он публикует свежевыявленные проблемы безопасности и возможные способы их решения. Информация о подписке доступна на *http://www.securityfocus.com/archive*.

- Beyond Linux From Scratch

  Книга Beyond Linux From Scratch описывает установку широкого спектра программного обеспечения, не входящего в круг рассмотрения книги LFS. Проект BLFS расположен по адресу *http://www.linuxfromscratch.org/blfs/*.

- LFS Hints

LFS Hints - это коллекция образовательных документов, созданных добровольцами из сообщества LFS. Советы доступны здесь: *http://www.linuxfromscratch.org/hints/list.html*.

- Списки рассылки

  Существуют несколько списков рассылки LFS, на которые Вы бы могли подписаться, если Вам нужна помощь, Вы хотите оставаться в курсе последних разработок или желаете помочь проекту. Подробнее здесь: Глава 1 - Списки рассылки.

- The Linux Documentation Project

  Цель The Linux Documentation Project (TLDP) - собрать документацию по всем аспектам использования и настройки Linux. Сайт TLDP содержит большую коллекцию HOWTO, руководств и справочных страниц. Он расположен по адресу *http://www.tldp.org/*.

Часть IV. Приложения

Приложение A. Acronyms and Terms

ABI Application Binary Interface
ALFS Automated Linux From Scratch
ALSA Advanced Linux Sound Architecture
API Application Programming Interface
ASCII American Standard Code for Information Interchange
BIOS Basic Input/Output System
BLFS Beyond Linux From Scratch
BSD Berkeley Software Distribution
chroot change root
CMOS Complementary Metal Oxide Semiconductor
COS Class Of Service
CPU Central Processing Unit
CRC Cyclic Redundancy Check
CVS Concurrent Versions System
DHCP Dynamic Host Configuration Protocol
DNS Domain Name Service
EGA Enhanced Graphics Adapter
ELF Executable and Linkable Format
EOF End of File
EQN equation
EVMS Enterprise Volume Management System
ext2 second extended file system
ext3 third extended file system
ext4 fourth extended file system
FAQ Frequently Asked Questions
FHS Filesystem Hierarchy Standard
FIFO First-In, First Out
FQDN Fully Qualified Domain Name
FTP File Transfer Protocol
GB Gigabytes
GCC GNU Compiler Collection
GID Group Identifier
GMT Greenwich Mean Time
GPG GNU Privacy Guard
HTML Hypertext Markup Language
IDE Integrated Drive Electronics
IEEE Institute of Electrical and Electronic Engineers

IO Input/Output
IP Internet Protocol
IPC Inter-Process Communication
IRC Internet Relay Chat
ISO International Organization for Standardization
ISP Internet Service Provider
KB Kilobytes
LED Light Emitting Diode
LFS Linux From Scratch
LSB Linux Standard Base
MB Megabytes
MBR Master Boot Record
MD5 Message Digest 5
NIC Network Interface Card
NLS Native Language Support
NNTP Network News Transport Protocol
NPTL Native POSIX Threading Library
OSS Open Sound System
PCH Pre-Compiled Headers
PCRE Perl Compatible Regular Expression
PID Process Identifier
PLFS Pure Linux From Scratch
PTY pseudo terminal
QA Quality Assurance
QOS Quality Of Service
RAM Random Access Memory
RPC Remote Procedure Call
RTC Real Time Clock
SBU Standard Build Unit
SCO The Santa Cruz Operation
SGR Select Graphic Rendition
SHA1 Secure-Hash Algorithm 1
SMP Symmetric Multi-Processor
TLDP The Linux Documentation Project
TFTP Trivial File Transfer Protocol
TLS Thread-Local Storage
UID User Identifier
umask user file-creation mask

USB Universal Serial Bus
UTC Coordinated Universal Time
UUID Universally Unique Identifier
VC Virtual Console
VGA Video Graphics Array
VT Virtual Terminal

Приложение B. Acknowledgments

We would like to thank the following people and organizations for their contributions to the Linux From Scratch Project.

- *Gerard Beekmans* <gerard@linuxfromscratch.org> – LFS Creator, LFS Project Leader
- *Matthew Burgess* <matthew@linuxfromscratch.org> – LFS Project Leader, LFS Technical Writer/Editor
- *Bruce Dubbs* <bdubbs@linuxfromscratch.org> – LFS Release Manager, LFS Technical Writer/Editor
- *Jim Gifford* <jim@linuxfromscratch.org> – CLFS Project Co-Leader
- *Bryan Kadzban* <bryan@linuxfromscratch.org> – LFS Technical Writer
- *Randy McMurchy* <randy@linuxfromscratch.org> – BLFS Project Leader, LFS Editor
- *DJ Lucas* <dj@linuxfromscratch.org> – LFS and BLFS Editor
- *Ken Moffat* <ken@linuxfromscratch.org> – LFS and CLFS Editor
- *Ryan Oliver* <ryan@linuxfromscratch.org> – CLFS Project Co-Leader
- Countless other people on the various LFS and BLFS mailing lists who helped make this book possible by giving their suggestions, testing the book, and submitting bug reports, instructions, and their experiences with installing various packages.

Translators

- *Manuel Canales Esparcia* <macana@macana-es.com> – Spanish LFS translation project
- *Johan Lenglet* <johan@linuxfromscratch.org> – French LFS translation project
- *Anderson Lizardo* <lizardo@linuxfromscratch.org> – Portuguese LFS translation project
- *Thomas Reitelbach* <tr@erdfunkstelle.de> – German LFS translation project

Mirror Maintainers

North American Mirrors

- *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

South American Mirrors

- *Manuel Canales Esparcia* <manuel@linuxfromscratch.org> – lfsmirror.lfs-es.info mirror
- *Luis Falcon* <Luis Falcon> – torredehanoi.org mirror

European Mirrors

- *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 Baqué* <baque@cict.fr> – lfs.cict.fr mirror
- *Vitaly Chekasin* <gyouja@pilgrims.ru> – lfs.pilgrims.ru mirror
- *Benjamin Heil* <kontakt@wankoo.org> – lfs.wankoo.org mirror

Asian Mirrors

- *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

Australian Mirrors

- *Jason Andrade* <jason@dstc.edu.au> – au.linuxfromscratch.org mirror

Former Project Team Members

- *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
- *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
- Mark Hymers
- Seth W. Klein – FAQ maintainer
- *Nicholas Leippe* <nicholas@linuxfromscratch.org> – Wiki Maintainer

- *Anderson Lizardo* <lizardo@linuxfromscratch.org> – Website Backend-Scripts Maintainer
- *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
- *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

Приложение C. Dependencies

Every package built in LFS relies on one or more other packages in order to build and install properly. Some packages even participate in circular dependencies, that is, the first package depends on the second which in turn depends on the first. Because of these dependencies, the order in which packages are built in LFS is very important. The purpose of this page is to document the dependencies of each package built in LFS.

For each package we build, we have listed three, and sometimes four, types of dependencies. The first lists what other packages need to be available in order to compile and install the package in question. The second lists what packages, in addition to those on the first list, need to be available in order to run the test suites. The third list of dependencies are packages that require this package to be built and installed in its final location before they are built and installed. In most cases, this is because these packages will hardcode paths to binaries within their scripts. If not built in a certain order, this could result in paths of /tools/bin/[binary] being placed inside scripts installed to the final system. This is obviously not desirable.

The last list of dependencies are optional packages that are not addressed in LFS, but could be useful to the user. These packages may have additional mandatory or optional dependencies of their own. For these dependencies, the recommeded practice is to install them after completion of the LFS book and then go back an rebuild the LFS package. In several cases, reinstallation is addressed in BLFS.

Autoconf

Для установки необходимы: Bash, Coreutils, Grep, M4, Make, Perl, Sed, and Texinfo

Для тестов необходимы: Automake, Diffutils, Findutils, GCC, and Libtool

Необходимо установить перед: Automake

Необязательные зависимости: Emacs

Automake

Для установки необходимы: Autoconf, Bash, Coreutils, Gettext, Grep, M4, Make, Perl, Sed, and Texinfo

Для тестов необходимы: Binutils, Bison, Bzip2, DejaGNU, Diffutils, Expect, Findutils, Flex, GCC, Gettext, Gzip, Libtool, and Tar.

Необходимо установить перед: None

Необязательные зависимости: None

Bash

Для установки необходимы: Bash, Binutils, Bison, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Make, Ncurses, Patch, Readline, Sed, and Texinfo
Для тестов необходимы: None
Необходимо установить перед: None
Необязательные зависимости: Xorg

Binutils

Для установки необходимы: Bash, Binutils, Coreutils, Diffutils, File, Gawk, GCC, Glibc, Grep, Make, Perl, Sed, Texinfo and Zlib
Для тестов необходимы: DejaGNU and Expect
Необходимо установить перед: None
Необязательные зависимости: None

Bison

Для установки необходимы: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, M4, Make, and Sed
Для тестов необходимы: Diffutils and Findutils
Необходимо установить перед: Flex, Kbd, and Tar
Необязательные зависимости: Doxygen (test suite)

Bzip2

Для установки необходимы: Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Make, and Patch
Для тестов необходимы: None
Необходимо установить перед: None
Необязательные зависимости: None

Coreutils

Для установки необходимы: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, GMP, Grep, Make, Patch, Perl, Sed, and Texinfo
Для тестов необходимы: Diffutils, E2fsprogs, Findutils, and Util-linux
Необходимо установить перед: Bash, Diffutils, Findutils, Man-DB, and Udev
Необязательные зависимости: Perl Expect and IO:Tty modules (for test suite)

DejaGNU

Для установки необходимы: Bash, Coreutils, Diffutils, GCC, Grep, Make, and Sed
Для тестов необходимы: No test suite available
Необходимо установить перед: None
Необязательные зависимости: None

Diffutils

Для установки необходимы: Bash, Binutils, Coreutils, Gawk, GCC, Gettext, Glibc, Grep, Make, Sed, and Texinfo
Для тестов необходимы: Diffutils, Perl
Необходимо установить перед: None
Необязательные зависимости: None

Expect

Для установки необходимы: Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Grep, Make, Patch, Sed, and Tcl
Для тестов необходимы: None
Необходимо установить перед: None
Необязательные зависимости: None

E2fsprogs

Для установки необходимы: Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Gzip, Make, Pkg-config, Sed, Texinfo, and Util-linux
Для тестов необходимы: None
Необходимо установить перед: None
Необязательные зависимости: None

File

Для установки необходимы: Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Make, Sed, and Zlib
Для тестов необходимы: None
Необходимо установить перед: None
Необязательные зависимости: None

Findutils

Для установки необходимы: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, Make, Sed, and Texinfo
Для тестов необходимы: DejaGNU, Diffutils, and Expect
Необходимо установить перед: None
Необязательные зависимости: None

Flex

Для установки необходимы: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, M4, Make, Patch, Sed, and Texinfo
Для тестов необходимы: Bison and Gawk
Необходимо установить перед: IPRoute2, Kbd, and Man-DB
Необязательные зависимости: None

Gawk

Для установки необходимы: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, Make, Patch, Sed and, Texinfo
Для тестов необходимы: Diffutils
Необходимо установить перед: None
Необязательные зависимости: None

Gcc

Для установки необходимы: Bash, Binutils, Coreutils, Diffutils, Findutils, Gawk, GCC, Gettext, Glibc, GMP, Grep, M4, Make, MPC, MPFR, Patch, Perl, Sed, Tar, and Texinfo
Для тестов необходимы: DejaGNU and Expect
Необходимо установить перед: None
Необязательные зависимости: *CLooG-PPL*, *GNAT* and *PPL*

GDBM

Для установки необходимы: Bash, Binutils, Coreutils, Diffutils, GCC, Grep, Make, and Sed
Для тестов необходимы: None
Необходимо установить перед: None
Необязательные зависимости: None

Gettext

Для установки необходимы: Bash, Binutils, Coreutils, Gawk, GCC, Glibc, Grep, Make, Sed, and Texinfo
Для тестов необходимы: Diffutils, Perl, and Tcl
Необходимо установить перед: Automake
Необязательные зависимости: None

Glibc

Для установки необходимы: Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Gettext, Grep, Gzip, Linux API Headers, Make, Perl, Sed, and Texinfo
Для тестов необходимы: File
Необходимо установить перед: None
Необязательные зависимости: None

GMP

Для установки необходимы: Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, M4, Make, Sed and Texinfo
Для тестов необходимы: None
Необходимо установить перед: MPFR, GCC
Необязательные зависимости: None

Grep

Для установки необходимы: Bash, Binutils, Coreutils, Diffutils, GCC, Gettext, Glibc, Grep, Make, Patch, Sed, and Texinfo
Для тестов необходимы: Gawk
Необходимо установить перед: Man-DB
Необязательные зависимости: Pcre, Xorg, and CUPS

Groff

Для установки необходимы: Bash, Binutils, Bison, Coreutils, Gawk, GCC, Glibc, Grep, Make, Patch, Sed, and Texinfo
Для тестов необходимы: No test suite available
Необходимо установить перед: Man-DB and Perl
Необязательные зависимости: GPL Ghostscript

GRUB

Для установки необходимы: Bash, Binutils, Bison, Coreutils, Diffutils, GCC, Gettext, Glibc, Grep, Make, Ncurses, Sed, and Texinfo
Для тестов необходимы: None
Необходимо установить перед: None
Необязательные зависимости: None

Gzip

Для установки необходимы: Bash, Binutils, Coreutils, GCC, Glibc, Grep, Make, Sed, and Texinfo
Для тестов необходимы: Diffutils
Необходимо установить перед: Man-DB
Необязательные зависимости: None

Iana-Etc

Для установки необходимы: Coreutils, Gawk, and Make
Для тестов необходимы: No test suite available
Необходимо установить перед: Perl
Необязательные зависимости: None

Inetutils

Для установки необходимы: Bash, Binutils, Coreutils, GCC, Glibc, Grep, Make, Ncurses, Patch, Sed, Texinfo, and Zlib
Для тестов необходимы: No test suite available
Необходимо установить перед: Tar
Необязательные зависимости: None

IProute2

Для установки необходимы: Bash, Bison, Coreutils, Flex, GCC, Glibc, Make, and Linux API Headers
Для тестов необходимы: No test suite available
Необходимо установить перед: None
Необязательные зависимости: None

Kbd

Для установки необходимы: Bash, Binutils, Bison, Coreutils, Flex, GCC, Gettext, Glibc, Gzip, Make, Patch, and Sed
Для тестов необходимы: No test suite available
Необходимо установить перед: None
Необязательные зависимости: None

Less

Для установки необходимы: Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Grep, Make, Ncurses, and Sed
Для тестов необходимы: No test suite available
Необходимо установить перед: None
Необязательные зависимости: Pcre

Libtool

Для установки необходимы: Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Make, Sed, and Texinfo
Для тестов необходимы: Findutils
Необходимо установить перед: None
Необязательные зависимости: None

Linux Kernel

Для установки необходимы: Bash, Binutils, Coreutils, Diffutils, Findutils, GCC, Glibc, Grep, Gzip, Make, Module-Init-Tools, Ncurses, Perl, and Sed
Для тестов необходимы: No test suite available
Необходимо установить перед: None
Необязательные зависимости: None

M4

Для установки необходимы: Bash, Binutils, Coreutils, GCC, Glibc, Grep, Make, Sed, and Texinfo
Для тестов необходимы: Diffutils
Необходимо установить перед: Autoconf and Bison
Необязательные зависимости: libsigsegv

Make

Для установки необходимы: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, Make, Sed, and Texinfo
Для тестов необходимы: Perl and Procps
Необходимо установить перед: None
Необязательные зависимости: None

Man-DB

Для установки необходимы: Bash, Binutils, Bzip2, Coreutils, Flex, GCC, GDBM, Gettext, Glibc, Grep, Groff, Gzip, Less, Make, Sed, and Xz
Для тестов необходимы: Not run. Requires Man-DB test suite package
Необходимо установить перед: None
Необязательные зависимости: None

Man-Pages

Для установки необходимы: Bash, Coreutils, and Make
Для тестов необходимы: No test suite available
Необходимо установить перед: None
Необязательные зависимости: None

Module-Init-Tools

Для установки необходимы: Bash, Binutils, Coreutils, Findutils, GCC, Glibc, Grep, Make, Patch, Sed, and Zlib
Для тестов необходимы: Diffutils, File, Gawk, and Gzip
Необходимо установить перед: None
Необязательные зависимости: None

MPC

Для установки необходимы: Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, GMP, Make, MPFR, Sed and Texinfo
Для тестов необходимы: None
Необходимо установить перед: GCC
Необязательные зависимости: None

MPFR

Для установки необходимы: Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, GMP, Make, Sed and Texinfo
Для тестов необходимы: None
Необходимо установить перед: GCC
Необязательные зависимости: None

Ncurses

Для установки необходимы: Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Make, Patch, and Sed
Для тестов необходимы: No test suite available
Необходимо установить перед: Bash, GRUB, Inetutils, Less, Procps, Psmisc, Readline, Texinfo, Util-linux, and Vim
Необязательные зависимости: None

Patch

Для установки необходимы: Bash, Binutils, Coreutils, GCC, Glibc, Grep, Make, and Sed
Для тестов необходимы: No test suite available
Необходимо установить перед: None
Необязательные зависимости: Ed

Perl

Для установки необходимы: Bash, Binutils, Coreutils, Gawk, GCC, GDBM, Glibc, Grep, Groff, Make, Sed, and Zlib
Для тестов необходимы: Iana-Etc and Procps
Необходимо установить перед: Autoconf
Необязательные зависимости: None

Pkg-config

Для установки необходимы: Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Make, and Sed
Для тестов необходимы: None
Необходимо установить перед: None
Необязательные зависимости: None

Procps

Для установки необходимы: Bash, Binutils, Coreutils, GCC, Glibc, Make, and Ncurses
Для тестов необходимы: No test suite available
Необходимо установить перед: None
Необязательные зависимости: None

Psmisc

Для установки необходимы: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, Make, Ncurses, and Sed
Для тестов необходимы: No test suite available
Необходимо установить перед: None
Необязательные зависимости: None

Readline

Для установки необходимы: Bash, Binutils, Coreutils, Gawk, GCC, Glibc, Grep, Make, Ncurses, Patch, Sed, and Texinfo
Для тестов необходимы: No test suite available
Необходимо установить перед: Bash
Необязательные зависимости: None

Sed

Для установки необходимы: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, Make, Sed, and Texinfo
Для тестов необходимы: Diffutils and Gawk
Необходимо установить перед: E2fsprogs, File, Libtool, and Shadow
Необязательные зависимости: Cracklib

Shadow

Для установки необходимы: Bash, Binutils, Coreutils, Diffutils, Findutils, Gawk, GCC, Gettext, Glibc, Grep, Make, and Sed
Для тестов необходимы: No test suite available
Необходимо установить перед: None
Необязательные зависимости: None

Sysklogd

Для установки необходимы: Binutils, Coreutils, GCC, Glibc, Make, and Patch
Для тестов необходимы: No test suite available
Необходимо установить перед: None
Необязательные зависимости: None

Sysvinit

Для установки необходимы: Binutils, Coreutils, GCC, Glibc, Make, and Sed
Для тестов необходимы: No test suite available
Необходимо установить перед: None
Необязательные зависимости: None

Tar

Для установки необходимы: Bash, Binutils, Bison, Coreutils, GCC, Gettext, Glibc, Grep, Inetutils, Make, Sed, and Texinfo
Для тестов необходимы: Autoconf, Diffutils, Findutils, Gawk, and Gzip
Необходимо установить перед: None
Необязательные зависимости: None

Tcl

Для установки необходимы: Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Grep, Make, and Sed
Для тестов необходимы: None
Необходимо установить перед: None
Необязательные зависимости: None

Texinfo

Для установки необходимы: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, Make, Ncurses, Patch, and Sed
Для тестов необходимы: None
Необходимо установить перед: None
Необязательные зависимости: None

Udev

Для установки необходимы: Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Make, and Sed
Для тестов необходимы: No test suite available
Необходимо установить перед: None
Необязательные зависимости: None

Util-linux

Для установки необходимы: Bash, Binutils, Coreutils, Diffutils, Findutils, Gawk, GCC, Gettext, Glibc, Grep, Make, Ncurses, Sed, and Zlib
Для тестов необходимы: No test suite available
Необходимо установить перед: None
Необязательные зависимости: None

Vim

Для установки необходимы: Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Grep, Make, Ncurses, and Sed
Для тестов необходимы: None
Необходимо установить перед: None
Необязательные зависимости: Xorg, GTK+2, LessTif, Python, Tcl, Ruby, and GPM

Xz

Для установки необходимы: Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, and Make.
Для тестов необходимы: None
Необходимо установить перед: Man-DB
Необязательные зависимости: None

Zlib

Для установки необходимы: Bash, Binutils, Coreutils, GCC, Glibc, Make, and Sed
Для тестов необходимы: None
Необходимо установить перед: File, Module-Init-Tools, Perl, and Util-linux
Необязательные зависимости: None

Приложение D. Boot and sysconfig scripts version-20100627

The scripts in this appendix are listed by the directory where they normally reside. The order is `/etc/rc.d/init.d`, `/etc/sysconfig`, `/etc/sysconfig/network-devices`, and `/etc/sysconfig/network-devices/services`. Within each section, the files are listed in the order they are normally called.

D.1. /etc/rc.d/init.d/rc

The `rc` script is the first script called by init and initiates the boot process.

```
#!/bin/sh
########################################################################
# Begin $rc_base/init.d/rc
#
# Description : Main Run Level Control Script
#
# Authors     : Gerard Beekmans  - gerard@linuxfromscratch.org
#
# Version     : 00.00
#
# Notes       :
#
########################################################################

. /etc/sysconfig/rc
. ${rc_functions}

# This sets a few default terminal options.
stty sane

# 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 ${rc_base}/rc${runlevel}.d ]; then
    boot_mesg "${rc_base}/rc${runlevel}.d does not exist." ${WARNING}
    boot_mesg_flush
    exit 1
fi

# Attempt to stop all service started by previous runlevel,
# and killed in this runlevel
if [ "${previous}" != "N" ]; then
    for i in $(ls -v ${rc_base}/rc${runlevel}.d/K* 2> /dev/null)
    do
        check_script_status
```

```
        suffix=${i#$rc_base/rc$runlevel.d/K[0-9][0-9]}
        prev_start=$rc_base/rc$previous.d/S[0-9][0-9]$suffix
        sysinit_start=$rc_base/rcsysinit.d/S[0-9][0-9]$suffix

        if [ "${runlevel}" != "0" ] && [ "${runlevel}" != "6" ]; then
            if [ ! -f ${prev_start} ] && [ ! -f ${sysinit_start} ]; then
                boot_mesg -n "WARNING:\n\n${i} can't be" ${WARNING}
                boot_mesg -n " executed because it was not"
                boot_mesg -n " not started in the previous"
                boot_mesg -n " runlevel (${previous})."
                boot_mesg "" ${NORMAL}
                boot_mesg_flush
                continue
            fi
        fi
        ${i} stop
        error_value=${?}

        if [ "${error_value}" != "0" ]; then
            print_error_msg
        fi
    done
fi

#Start all functions in this runlevel
for i in $( ls -v ${rc_base}/rc${runlevel}.d/S* 2> /dev/null)
do
    if [ "${previous}" != "N" ]; then
        suffix=${i#$rc_base/rc$runlevel.d/S[0-9][0-9]}
        stop=$rc_base/rc$runlevel.d/K[0-9][0-9]$suffix
        prev_start=$rc_base/rc$previous.d/S[0-9][0-9]$suffix

        [ -f ${prev_start} ] && [ ! -f ${stop} ] && continue
    fi

    check_script_status

    case ${runlevel} in
        0|6)
            ${i} stop
            ;;
        *)
            ${i} start
            ;;
    esac
    error_value=${?}

    if [ "${error_value}" != "0" ]; then
        print_error_msg
    fi
done

# End $rc_base/init.d/rc
```

D.2. /etc/rc.d/init.d/functions

```
#!/bin/sh
########################################################################
# Begin $rc_base/init.d/functions
```

```
#
# Description : Run Level Control Functions
#
# Authors     : Gerard Beekmans - gerard@linuxfromscratch.org
#
# Version     : 00.00
#
# Notes       : With code based on Matthias Benkmann's simpleinit-msb
#       http://winterdrache.de/linux/newboot/index.html
#
########################################################################

## Environmental setup
# Setup default values for environment
umask 022
export PATH="/bin:/usr/bin:/sbin:/usr/sbin"

# Signal sent to running processes to refresh their configuration
RELOADSIG="HUP"

# Number of seconds between STOPSIG and FALLBACK when stopping processes
KILLDELAY="3"

## Screen Dimensions
# Find current screen size
if [ -z "${COLUMNS}" ]; then
    COLUMNS=$(stty size)
    COLUMNS=${COLUMNS##* }
fi

# When using remote connections, such as a serial port, stty size returns 0
if [ "${COLUMNS}" = "0" ]; then
    COLUMNS=80
fi

## Measurements for positioning result messages
COL=$((${COLUMNS} - 8))
WCOL=$((${COL} - 2))

## Provide an echo that supports -e and -n
# If formatting is needed, $ECHO should be used
case "`echo -e -n test`" in
    -[en]*)
        ECHO=/bin/echo
        ;;
    *)
        ECHO=echo
        ;;
esac

## Set Cursor Position Commands, used via $ECHO
SET_COL="\\033[${COL}G"      # at the $COL char
SET_WCOL="\\033[${WCOL}G"    # at the $WCOL char
CURS_UP="\\033[1A\\033[0G"   # Up one line, at the 0'th char

## Set color commands, used via $ECHO
# Please consult `man console_codes for more information
# under the "ECMA-48 Set Graphics Rendition" section
#
# Warning: when switching from a 8bit to a 9bit font,
# the linux console will reinterpret the bold (1;) to
```

```
# the top 256 glyphs of the 9bit font.  This does
# not affect framebuffer consoles
NORMAL="\\033[0;39m"         # Standard console grey
SUCCESS="\\033[1;32m"        # Success is green
WARNING="\\033[1;33m"        # Warnings are yellow
FAILURE="\\033[1;31m"        # Failures are red
INFO="\\033[1;36m"           # Information is light cyan
BRACKET="\\033[1;34m"        # Brackets are blue

STRING_LENGTH="0"   # the length of the current message

#*************************************************************************
# Function - boot_mesg()
#
# Purpose:      Sending information from bootup scripts to the console
#
# Inputs:       $1 is the message
#               $2 is the colorcode for the console
#
# Outputs:      Standard Output
#
# Dependencies: - sed for parsing strings.
#             - grep for counting string length.
#
# Todo:
#*************************************************************************
boot_mesg()
{
    local ECHOPARM=""

    while true
    do
        case "${1}" in
            -n)
                ECHOPARM=" -n "
                shift 1
                ;;
            -*)
                echo "Unknown Option: ${1}"
                return 1
                ;;
            *)
                break
                ;;
        esac
    done

    ## Figure out the length of what is to be printed to be used
    ## for warning messages.
    STRING_LENGTH=$((${#1} + 1))

    # Print the message to the screen
    ${ECHO} ${ECHOPARM} -e "${2}${1}"

}

boot_mesg_flush()
{
    # Reset STRING_LENGTH for next message
    STRING_LENGTH="0"
}
```

```
boot_log()
{
    # Left in for backwards compatibility
    :
}

echo_ok()
{
    ${ECHO} -n -e "${CURS_UP}${SET_COL}${BRACKET}[${SUCCESS}  OK  ${BRACKET}]"
    ${ECHO} -e "${NORMAL}"
        boot_mesg_flush
}

echo_failure()
{
    ${ECHO} -n -e "${CURS_UP}${SET_COL}${BRACKET}[${FAILURE} FAIL ${BRACKET}]"
    ${ECHO} -e "${NORMAL}"
        boot_mesg_flush
}

echo_warning()
{
    ${ECHO} -n -e "${CURS_UP}${SET_COL}${BRACKET}[${WARNING} WARN ${BRACKET}]"
    ${ECHO} -e "${NORMAL}"
        boot_mesg_flush
}

print_error_msg()
{
    echo_failure
    # $i is inherited by the rc script
    boot_mesg -n "FAILURE:\n\nYou should not be reading this error message.\n\n" ${FAILURE}
    boot_mesg -n " It means that an unforeseen error took"
    boot_mesg -n " place in ${i}, which exited with a return value of"
    boot_mesg " ${error_value}.\n"
    boot_mesg_flush
    boot_mesg -n "If you're able to track this"
    boot_mesg -n " error down to a bug in one of the files provided by"
    boot_mesg -n " the LFS book, please be so kind to inform us at"
    boot_mesg " lfs-dev@linuxfromscratch.org.\n"
    boot_mesg_flush
    boot_mesg -n "Press Enter to continue..." ${INFO}
    boot_mesg "" ${NORMAL}
    read ENTER
}

check_script_status()
{
    # $i is inherited by the rc script
    if [ ! -f ${i} ]; then
        boot_mesg "${i} is not a valid symlink." ${WARNING}
        echo_warning
        continue
    fi

    if [ ! -x ${i} ]; then
        boot_mesg "${i} is not executable, skipping." ${WARNING}
        echo_warning
        continue
    fi
```

```
}

evaluate_retval()
{
    error_value="${?}"

    if [ ${error_value} = 0 ]; then
        echo_ok
    else
        echo_failure
    fi

    # This prevents the 'An Unexpected Error Has Occurred' from trivial
    # errors.
    return 0
}

print_status()
{
    if [ "${#}" = "0" ]; then
        echo "Usage: ${0} {success|warning|failure}"
        return 1
    fi

    case "${1}" in

        success)
            echo_ok
            ;;

        warning)
            # Leave this extra case in because old scripts
            # may call it this way.
            case "${2}" in
                running)
                    ${ECHO} -e -n "${CURS_UP}"
                    ${ECHO} -e -n "\\033[${STRING_LENGTH}G   "
                    boot_mesg "Already running." ${WARNING}
                    echo_warning
                    ;;
                not_running)
                    ${ECHO} -e -n "${CURS_UP}"
                    ${ECHO} -e -n "\\033[${STRING_LENGTH}G   "
                    boot_mesg "Not running." ${WARNING}
                    echo_warning
                    ;;
                not_available)
                    ${ECHO} -e -n "${CURS_UP}"
                    ${ECHO} -e -n "\\033[${STRING_LENGTH}G   "
                    boot_mesg "Not available." ${WARNING}
                    echo_warning
                    ;;
                *)
                    # This is how it is supposed to
                    # be called
                    echo_warning
                    ;;
            esac
        ;;

        failure)
```

```
            echo_failure
        ;;

    esac

}

reloadproc()
{
    local pidfile=""
    local failure=0

    while true
    do
        case "${1}" in
            -p)
                pidfile="${2}"
                shift 2
                ;;
            -*)
                log_failure_msg "Unknown Option: ${1}"
                return 2
                ;;
            *)
                break
                ;;
        esac
    done

    if [ "${#}" -lt "1" ]; then
        log_failure_msg "Usage: reloadproc [-p pidfile] pathname"
        return 2
    fi

    # This will ensure compatibility with previous LFS Bootscripts
    if [ -n "${PIDFILE}" ];    then
        pidfile="${PIDFILE}"
    fi

    # Is the process running?
    if [ -z "${pidfile}" ];    then
        pidofproc -s "${1}"
    else
        pidofproc -s -p "${pidfile}" "${1}"
    fi

    # Warn about stale pid file
    if [ "$?" = 1 ]; then
        boot_mesg -n "Removing stale pid file: ${pidfile}. " ${WARNING}
        rm -f "${pidfile}"
    fi

    if [ -n "${pidlist}" ];    then
        for pid in ${pidlist}
        do
            kill -"${RELOADSIG}" "${pid}" || failure="1"
        done

        (exit ${failure})
        evaluate_retval
```

```
    else
        boot_mesg "Process ${1} not running." ${WARNING}
        echo_warning
    fi
}

statusproc()
{
    local pidfile=""
    local base=""
    local ret=""

    while true
    do
        case "${1}" in
            -p)
                pidfile="${2}"
                shift 2
                ;;
            -*)
                log_failure_msg "Unknown Option: ${1}"
                return 2
                ;;
            *)
                break
                ;;
        esac
    done

    if [ "${#}" != "1" ]; then
        shift 1
        log_failure_msg "Usage: statusproc [-p pidfile] pathname"
        return 2
    fi

    # Get the process basename
    base="${1##*/}"

    # This will ensure compatibility with previous LFS Bootscripts
    if [ -n "${PIDFILE}" ];    then
        pidfile="${PIDFILE}"
    fi

    # Is the process running?
    if [ -z "${pidfile}" ];    then
        pidofproc -s "${1}"
    else
        pidofproc -s -p "${pidfile}" "${1}"
    fi

    # Store the return status
    ret=$?

    if [ -n "${pidlist}" ];    then
        ${ECHO} -e "${INFO}${base} is running with Process"\
            "ID(s) ${pidlist}.${NORMAL}"
    else
        if [ -n "${base}" -a -e "/var/run/${base}.pid" ]; then
            ${ECHO} -e "${WARNING}${1} is not running but"\
                "/var/run/${base}.pid exists.${NORMAL}"
        else
```

```
            if [ -n "${pidfile}" -a -e "${pidfile}" ]; then
                ${ECHO} -e "${WARNING}${1} is not running"\
                    "but ${pidfile} exists.${NORMAL}"
            else
                ${ECHO} -e "${INFO}${1} is not running.${NORMAL}"
            fi
        fi
    fi

    # Return the status from pidofproc
    return $ret
}

# The below functions are documented in the LSB-generic 2.1.0

#****************************************************************************
# Function - pidofproc [-s] [-p pidfile] pathname
#
# Purpose: This function returns one or more pid(s) for a particular daemon
#
# Inputs: -p pidfile, use the specified pidfile instead of pidof
#         pathname, path to the specified program
#
# Outputs: return 0 - Success, pid's in stdout
#          return 1 - Program is dead, pidfile exists
#          return 2 - Invalid or excessive number of arguments,
#                     warning in stdout
#          return 3 - Program is not running
#
# Dependencies: pidof, echo, head
#
# Todo: Remove dependency on head
#       This replaces getpids
#       Test changes to pidof
#
#****************************************************************************
pidofproc()
{
    local pidfile=""
    local lpids=""
    local silent=""
    pidlist=""
    while true
    do
        case "${1}" in
            -p)
                pidfile="${2}"
                shift 2
                ;;

            -s)
                # Added for legacy opperation of getpids
                # eliminates several '> /dev/null'
                silent="1"
                shift 1
                ;;
            -*)
                log_failure_msg "Unknown Option: ${1}"
                return 2
                ;;
            *)
```

```
                break
                ;;
        esac
    done

    if [ "${#}" != "1" ]; then
        shift 1
        log_failure_msg "Usage: pidofproc [-s] [-p pidfile] pathname"
        return 2
    fi

    if [ -n "${pidfile}" ]; then
        if [ ! -r "${pidfile}" ]; then
            return 3 # Program is not running
        fi

        lpids=`head -n 1 ${pidfile}`
        for pid in ${lpids}
        do
            if [ "${pid}" -ne "$$" -a "${pid}" -ne "${PPID}" ]; then
                kill -0 "${pid}" 2>/dev/null &&
                pidlist="${pidlist} ${pid}"
            fi

            if [ "${silent}" != "1" ]; then
                echo "${pidlist}"
            fi

            test -z "${pidlist}" &&
            # Program is dead, pidfile exists
            return 1
            # else
            return 0
        done

    else
        pidlist=`pidof -o $$ -o $PPID -x "$1"`
        if [ "${silent}" != "1" ]; then
            echo "${pidlist}"
        fi

        # Get provide correct running status
        if [ -n "${pidlist}" ]; then
            return 0
        else
            return 3
        fi

    fi

    if [ "$?" != "0" ]; then
        return 3 # Program is not running
    fi
}

#****************************************************************************
# Function - loadproc [-f] [-n nicelevel] [-p pidfile] pathname [args]
#
# Purpose: This runs the specified program as a daemon
#
# Inputs: -f, run the program even if it is already running
```

```
#         -n nicelevel, specifies a nice level. See nice(1).
#         -p pidfile, uses the specified pidfile
#         pathname, pathname to the specified program
#         args, arguments to pass to specified program
#
# Outputs: return 0 - Success
#          return 2 - Invalid of excessive number of arguments,
#                     warning in stdout
#          return 4 - Program or service status is unknown
#
# Dependencies: nice, rm
#
# Todo: LSB says this should be called start_daemon
#       LSB does not say that it should call evaluate_retval
#       It checks for PIDFILE, which is deprecated.
#         Will be removed after BLFS 6.0
#       loadproc returns 0 if program is already running, not LSB compliant
#
#***************************************************************************
loadproc()
{
    local pidfile=""
    local forcestart=""
    local nicelevel="10"

# This will ensure compatibility with previous LFS Bootscripts
    if [ -n "${PIDFILE}" ];    then
        pidfile="${PIDFILE}"
    fi

  while true
    do
        case "${1}" in
            -f)
                forcestart="1"
                shift 1
                ;;
            -n)
                nicelevel="${2}"
                shift 2
                ;;
            -p)
                pidfile="${2}"
                shift 2
                ;;
            -*)
                log_failure_msg "Unknown Option: ${1}"
                return 2 #invalid or excess argument(s)
                ;;
            *)
                break
                ;;
        esac
    done

    if [ "${#}" = "0" ]; then
        log_failure_msg "Usage: loadproc [-f] [-n nicelevel] [-p pidfile] pathname [args]"
        return 2 #invalid or excess argument(s)
    fi

    if [ -z "${forcestart}" ]; then
```

```
        if [ -z "${pidfile}" ];    then
            pidofproc -s "${1}"
        else
            pidofproc -s -p "${pidfile}" "${1}"
        fi

        case "${?}" in
            0)
                log_warning_msg "Unable to continue: ${1} is running"
                return 0 # 4
                ;;
            1)
                boot_mesg "Removing stale pid file: ${pidfile}" ${WARNING}
                rm -f "${pidfile}"
                ;;
            3)
                ;;
            *)
                log_failure_msg "Unknown error code from pidofproc: ${?}"
                return 4
                ;;
        esac
    fi

    nice -n "${nicelevel}" "${@}"
    evaluate_retval # This is "Probably" not LSB compliant,
#                         but required to be compatible with older bootscripts
    return 0
}

#****************************************************************************
# Function - killproc  [-p pidfile] pathname [signal]
#
# Purpose:
#
# Inputs: -p pidfile, uses the specified pidfile
#         pathname, pathname to the specified program
#         signal, send this signal to pathname
#
# Outputs: return 0 - Success
#          return 2 - Invalid of excessive number of arguments,
#                     warning in stdout
#          return 4 - Unknown Status
#
# Dependencies: kill, rm
#
# Todo: LSB does not say that it should call evaluate_retval
#       It checks for PIDFILE, which is deprecated.
#         Will be removed after BLFS 6.0
#
#****************************************************************************
killproc()
{
    local pidfile=""
    local killsig=TERM # default signal is SIGTERM
    pidlist=""

    # This will ensure compatibility with previous LFS Bootscripts
    if [ -n "${PIDFILE}" ];    then
        pidfile="${PIDFILE}"
    fi
```

```
    while true
    do
        case "${1}" in
            -p)
                pidfile="${2}"
                shift 2
                ;;
            -*)
                log_failure_msg "Unknown Option: ${1}"
                return 2
                ;;
            *)
                 break
                ;;
        esac
    done

    if [ "${#}" = "2" ]; then
        killsig="${2}"
    elif [ "${#}" != "1" ];    then
        shift 2
        log_failure_msg "Usage: killproc  [-p pidfile] pathname [signal]"
        return 2
    fi

    # Is the process running?
    if [ -z "${pidfile}" ];    then
        pidofproc -s "${1}"
    else
        pidofproc -s -p "${pidfile}" "${1}"
    fi

    # Remove stale pidfile
    if [ "$?" = 1 ]; then
        boot_mesg "Removing stale pid file: ${pidfile}." ${WARNING}
        rm -f "${pidfile}"
    fi

    # If running, send the signal
    if [ -n "${pidlist}" ]; then
    for pid in ${pidlist}
    do
        kill -${killsig} ${pid} 2>/dev/null

        # Wait up to 3 seconds, for ${pid} to terminate
        case "${killsig}" in
        TERM|SIGTERM|KILL|SIGKILL)
            # sleep in 1/10ths of seconds and
            # multiply KILLDELAY by 10
            local dtime="${KILLDELAY}0"
            while [ "${dtime}" != "0" ]
            do
                kill -0 ${pid} 2>/dev/null || break
                sleep 0.1
                dtime=$(( ${dtime} - 1))
            done
            # If ${pid} is still running, kill it
            kill -0 ${pid} 2>/dev/null && kill -KILL ${pid} 2>/dev/null
            ;;
        esac
```

```
    done

    # Check if the process is still running if we tried to stop it
    case "${killsig}" in
    TERM|SIGTERM|KILL|SIGKILL)
        if [ -z "${pidfile}" ];    then
            pidofproc -s "${1}"
        else
            pidofproc -s -p "${pidfile}" "${1}"
        fi

        # Program was terminated
        if [ "$?" != "0" ]; then
            # Remove the pidfile if necessary
            if [ -f "${pidfile}" ];    then
                rm -f "${pidfile}"
            fi
            echo_ok
            return 0
        else # Program is still running
            echo_failure
            return 4 # Unknown Status
        fi
        ;;
    *)
        # Just see if the kill returned successfully
        evaluate_retval
        ;;
    esac
    else # process not running
    print_status warning not_running
    fi
}


#****************************************************************************
# Function - log_success_msg "message"
#
# Purpose: Print a success message
#
# Inputs: $@ - Message
#
# Outputs: Text output to screen
#
# Dependencies: echo
#
# Todo: logging
#
#****************************************************************************
log_success_msg()
{
    ${ECHO} -n -e "${BOOTMESG_PREFIX}${@}"
    ${ECHO} -e "${SET_COL}""${BRACKET}""["${SUCCESS}""  OK  ""${BRACKET}""]""${NORMAL}"
    return 0
}

#****************************************************************************
# Function - log_failure_msg "message"
#
# Purpose: Print a failure message
#
```

```
# Inputs: $@ - Message
#
# Outputs: Text output to screen
#
# Dependencies: echo
#
# Todo: logging
#
#****************************************************************************
log_failure_msg() {
    ${ECHO} -n -e "${BOOTMESG_PREFIX}${@}"
    ${ECHO} -e "${SET_COL}""${BRACKET}""["${FAILURE}"" FAIL ""${BRACKET}""]""${NORMAL}"
    return 0
}

#****************************************************************************
# Function - log_warning_msg "message"
#
# Purpose: print a warning message
#
# Inputs: $@ - Message
#
# Outputs: Text output to screen
#
# Dependencies: echo
#
# Todo: logging
#
#****************************************************************************
log_warning_msg() {
    ${ECHO} -n -e "${BOOTMESG_PREFIX}${@}"
    ${ECHO} -e "${SET_COL}""${BRACKET}""["${WARNING}"" WARN ""${BRACKET}""]""${NORMAL}"
    return 0
}

# End $rc_base/init.d/functions
```

D.3. /etc/rc.d/init.d/mountkernfs

```
#!/bin/sh
########################################################################
# Begin $rc_base/init.d/mountkernfs
#
# Description : Mount proc and sysfs
#
# Authors     : Gerard Beekmans - gerard@linuxfromscratch.org
#
# Version     : 00.00
#
# Notes       :
#
########################################################################

. /etc/sysconfig/rc
. ${rc_functions}

case "${1}" in
    start)
        boot_mesg -n "Mounting kernel-based file systems:" ${INFO}
```

```
        if ! mountpoint /proc >/dev/null; then
            boot_mesg -n " /proc" ${NORMAL}
            mount -n /proc || failed=1
        fi

        if ! mountpoint /sys >/dev/null; then
            boot_mesg -n " /sys" ${NORMAL}
            mount -n /sys || failed=1
        fi

        boot_mesg "" ${NORMAL}

        (exit ${failed})
        evaluate_retval
        ;;

    *)
        echo "Usage: ${0} {start}"
        exit 1
        ;;
esac

# End $rc_base/init.d/mountkernfs
```

D.4. /etc/rc.d/init.d/consolelog

```
#!/bin/sh
# Begin $rc_base/init.d/consolelog

########################################################################
#
# Description : Set the kernel log level for the console
#
# Authors     : Dan Nicholson - dnicholson@linuxfromscratch.org
#
# Version     : 00.00
#
# Notes       : /proc must be mounted before this can run
#
########################################################################

. /etc/sysconfig/rc
. ${rc_functions}

# set the default loglevel
LOGLEVEL=7
if [ -r /etc/sysconfig/console ]; then
    . /etc/sysconfig/console
fi

case "${1}" in
    start)
        case "$LOGLEVEL" in
        [1-8])
            boot_mesg "Setting the console log level to ${LOGLEVEL}..."
            dmesg -n $LOGLEVEL
            evaluate_retval
            ;;
        *)
            boot_mesg "Console log level '${LOGLEVEL}' is invalid" ${FAILURE}
```

```
            echo_failure
            ;;
        esac
        ;;
    status)
        # Read the current value if possible
        if [ -r /proc/sys/kernel/printk ]; then
            read level line < /proc/sys/kernel/printk
        else
            boot_mesg "Can't read the current console log level" ${FAILURE}
            echo_failure
        fi

        # Print the value
        if [ -n "$level" ]; then
            ${ECHO} -e "${INFO}The current console log level" \
                "is ${level}${NORMAL}"
        fi
        ;;

    *)
        echo "Usage: ${0} {start|status}"
        exit 1
        ;;
esac

# End $rc_base/init.d/consolelog
```

D.5. /etc/rc.d/init.d/modules

```
#!/bin/sh
########################################################################
# Begin $rc_base/init.d/modules
#
# Description : Module auto-loading script
#
# Authors     : Zack Winkles
#
# Version     : 00.00
#
# Notes       :
#
########################################################################

. /etc/sysconfig/rc
. ${rc_functions}

# Assure that the kernel has module support.
[ -e /proc/ksyms -o -e /proc/modules ] || exit 0

case "${1}" in
    start)

        # Exit if there's no modules file or there are no
        # valid entries
        [ -r /etc/sysconfig/modules ] &&
            egrep -qv '^($|#)' /etc/sysconfig/modules ||
            exit 0

        boot_mesg -n "Loading modules:" ${INFO}
```

```
        # 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, making
            # sure to pass any arguments provided.
            modprobe ${module} ${args} >/dev/null

            # Print the module name if successful,
            # otherwise take note.
            if [ $? -eq 0 ]; then
                boot_mesg -n " ${module}" ${NORMAL}
            else
                failedmod="${failedmod} ${module}"
            fi
        done < /etc/sysconfig/modules

        boot_mesg "" ${NORMAL}
        # Print a message about successfully loaded
        # modules on the correct line.
        echo_ok

        # Print a failure message with a list of any
        # modules that may have failed to load.
        if [ -n "${failedmod}" ]; then
            boot_mesg "Failed to load modules:${failedmod}" ${FAILURE}
            echo_failure
        fi
        ;;
    *)
        echo "Usage: ${0} {start}"
        exit 1
        ;;
esac

# End $rc_base/init.d/modules
```

D.6. /etc/rc.d/init.d/udev

```
#!/bin/sh
########################################################################
# Begin $rc_base/init.d/udev
#
# Description : Udev cold-plugging script
#
# Authors     : Zack Winkles, Alexander E. Patrakov
#
# Version     : 00.02
#
# Notes       :
#
########################################################################

. /etc/sysconfig/rc
```

```
. ${rc_functions}

case "${1}" in
    start)
        boot_mesg "Populating /dev with device nodes..."
        if ! grep -q '[[:space:]]sysfs' /proc/mounts; then
            echo_failure
            boot_mesg -n "FAILURE:\n\nUnable to create" ${FAILURE}
            boot_mesg -n " devices without a SysFS filesystem"
            boot_mesg -n "\n\nAfter you press Enter, this system"
            boot_mesg -n " will be halted and powered off."
            boot_mesg -n "\n\nPress Enter to continue..." ${INFO}
            boot_mesg "" ${NORMAL}
            read ENTER
            /etc/rc.d/init.d/halt stop
        fi

        # Mount a temporary file system over /dev, so that any devices
        # made or removed during this boot don't affect the next one.
        # The reason we don't write to mtab is because we don't ever
        # want /dev to be unavailable (such as by `umount -a').
        if ! mountpoint /dev > /dev/null; then
            mount -n -t tmpfs tmpfs /dev -o mode=755
        fi
        if [ ${?} != 0 ]; then
            echo_failure
            boot_mesg -n "FAILURE:\n\nCannot mount a tmpfs" ${FAILURE}
            boot_mesg -n " onto /dev, this system will be halted."
            boot_mesg -n "\n\nAfter you press Enter, this system"
            boot_mesg -n " will be halted and powered off."
            boot_mesg -n "\n\nPress Enter to continue..." ${INFO}
            boot_mesg "" ${NORMAL}
            read ENTER
            /etc/rc.d/init.d/halt stop
        fi

        # Udev handles uevents itself, so we don't need to have
        # the kernel call out to any binary in response to them
        echo > /proc/sys/kernel/hotplug

        # Copy the only static device node that Udev >= 155 doesn't
        # handle to /dev
        cp -a /lib/udev/devices/null /dev

        # 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

        # Now wait for udevd to process the uevents we triggered
        /sbin/udevadm settle
        evaluate_retval

        ;;

    *)
        echo "Usage ${0} {start}"
        exit 1
```

```
        ;;
esac

# End $rc_base/init.d/udev
```

D.7. /etc/rc.d/init.d/swap

```
#!/bin/sh
########################################################################
# Begin $rc_base/init.d/swap
#
# Description : Swap Control Script
#
# Authors     : Gerard Beekmans - gerard@linuxfromscratch.org
#
# Version     : 00.00
#
# Notes       :
#
########################################################################

. /etc/sysconfig/rc
. ${rc_functions}

case "${1}" in
    start)
        boot_mesg "Activating all swap files/partitions..."
        swapon -a
        evaluate_retval
        ;;

    stop)
        boot_mesg "Deactivating all swap files/partitions..."
        swapoff -a
        evaluate_retval
        ;;

    restart)
        ${0} stop
        sleep 1
        ${0} start
        ;;

    status)
        boot_mesg "Retrieving swap status." ${INFO}
        echo_ok
        echo
        swapon -s
        ;;

    *)
        echo "Usage: ${0} {start|stop|restart|status}"
        exit 1
        ;;
esac

# End $rc_base/init.d/swap
```

D.8. /etc/rc.d/init.d/setclock

```
#!/bin/sh
########################################################################
# Begin $rc_base/init.d/setclock
#
# Description : Setting Linux Clock
#
# Authors     : Gerard Beekmans - gerard@linuxfromscratch.org
#
# Version     : 00.00
#
# Notes       :
#
########################################################################

. /etc/sysconfig/rc
. ${rc_functions}
. /etc/sysconfig/clock

case "${UTC}" in
    yes|true|1)
        CLOCKPARAMS="${CLOCKPARAMS} --utc"
        ;;

    no|false|0)
        CLOCKPARAMS="${CLOCKPARAMS} --localtime"
        ;;

esac

case ${1} in
    start)
        boot_mesg "Setting system clock..."
        hwclock --hctosys ${CLOCKPARAMS} >/dev/null
        evaluate_retval
        ;;

    stop)
        boot_mesg "Setting hardware clock..."
        hwclock --systohc ${CLOCKPARAMS} >/dev/null
        evaluate_retval
        ;;

    *)
        echo "Usage: ${0} {start|stop}"
        ;;

esac
```

D.9. /etc/rc.d/init.d/checkfs

```
#!/bin/sh
########################################################################
# Begin $rc_base/init.d/checkfs
#
# Description : File System Check
#
# Authors     : Gerard Beekmans - gerard@linuxfromscratch.org
#               A. Luebke - luebke@users.sourceforge.net
```

```
#
# Version     : 00.00
#
# Notes       :
#
# 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
#
########################################################################

. /etc/sysconfig/rc
. ${rc_functions}

case "${1}" in
    start)
        if [ -f /fastboot ]; then
            boot_mesg -n "/fastboot found, will not perform" ${INFO}
            boot_mesg " file system checks as requested."
            echo_ok
            exit 0
        fi

        boot_mesg "Mounting root file system in read-only mode..."
        mount -n -o remount,ro / >/dev/null
        evaluate_retval

        if [ ${?} != 0 ]; then
            echo_failure
            boot_mesg -n "FAILURE:\n\nCannot check root" ${FAILURE}
            boot_mesg -n " filesystem because it could not be mounted"
            boot_mesg -n " in read-only mode.\n\nAfter you"
            boot_mesg -n " press Enter, this system will be"
            boot_mesg -n " halted and powered off."
            boot_mesg -n "\n\nPress enter to continue..." ${INFO}
            boot_mesg "" ${NORMAL}
            read ENTER
            ${rc_base}/init.d/halt stop
        fi

        if [ -f /forcefsck ]; then
            boot_mesg -n "/forcefsck found, forcing file" ${INFO}
            boot_mesg " system checks as requested."
            echo_ok
            options="-f"
        else
            options=""
        fi

        boot_mesg "Checking file systems..."
        # Note: -a option used to be -p; but this fails e.g.
        # on fsck.minix
        fsck ${options} -a -A -C -T
```

```
        error_value=${?}

        if [ "${error_value}" = 0 ]; then
            echo_ok
        fi

        if [ "${error_value}" = 1 ]; then
            echo_warning
            boot_mesg -n "WARNING:\n\nFile system errors" ${WARNING}
            boot_mesg -n " were found and have been corrected."
            boot_mesg -n "  You may want to double-check that"
            boot_mesg -n " everything was fixed properly."
            boot_mesg "" ${NORMAL}
        fi

        if [ "${error_value}" = 2 -o "${error_value}" = 3 ]; then
            echo_warning
            boot_mesg -n "WARNING:\n\nFile system errors" ${WARNING}
            boot_mesg -n " were found and have been been"
            boot_mesg -n " corrected, but the nature of the"
            boot_mesg -n " errors require this system to be"
            boot_mesg -n " rebooted.\n\nAfter you press enter,"
            boot_mesg -n " this system will be rebooted"
            boot_mesg -n "\n\nPress Enter to continue..." ${INFO}
            boot_mesg "" ${NORMAL}
            read ENTER
            reboot -f
        fi

        if [ "${error_value}" -gt 3 -a "${error_value}" -lt 16 ]; then
            echo_failure
            boot_mesg -n "FAILURE:\n\nFile system errors" ${FAILURE}
            boot_mesg -n " were encountered that could not be"
            boot_mesg -n " fixed automatically.  This system"
            boot_mesg -n " cannot continue to boot and will"
            boot_mesg -n " therefore be halted until those"
            boot_mesg -n " errors are fixed manually by a"
            boot_mesg -n " System Administrator.\n\nAfter you"
            boot_mesg -n " press Enter, this system will be"
            boot_mesg -n " halted and powered off."
            boot_mesg -n "\n\nPress Enter to continue..." ${INFO}
            boot_mesg "" ${NORMAL}
            read ENTER
          ${rc_base}/init.d/halt stop
        fi

        if [ "${error_value}" -ge 16 ]; then
            echo_failure
            boot_mesg -n "FAILURE:\n\nUnexpected Failure" ${FAILURE}
            boot_mesg -n " running fsck.  Exited with error"
            boot_mesg -n " code: ${error_value}."
            boot_mesg "" ${NORMAL}
            exit ${error_value}
        fi
        ;;
    *)
        echo "Usage: ${0} {start}"
        exit 1
        ;;
esac
```

```
# End $rc_base/init.d/checkfs
```

D.10. /etc/rc.d/init.d/mountfs

```
#!/bin/sh
########################################################################
# Begin $rc_base/init.d/mountfs
#
# Description : File System Mount Script
#
# Authors     : Gerard Beekmans - gerard@linuxfromscratch.org
#
# Version     : 00.00
#
# Notes       :
#
########################################################################

. /etc/sysconfig/rc
. ${rc_functions}

case "${1}" in
    start)
        boot_mesg "Remounting root file system in read-write mode..."
        mount -n -o remount,rw / >/dev/null
        evaluate_retval

        # Remove fsck-related file system watermarks.
        rm -f /fastboot /forcefsck

        boot_mesg "Recording existing mounts in /etc/mtab..."
        > /etc/mtab
        mount -f / || failed=1
        mount -f /proc || failed=1
        mount -f /sys || failed=1
        (exit ${failed})
        evaluate_retval

        # This will mount all filesystems that do not have _netdev in
        # their option list.  _netdev denotes a network filesystem.
        boot_mesg "Mounting remaining file systems..."
        mount -a -O no_netdev >/dev/null
        evaluate_retval
        ;;

    stop)
        boot_mesg "Unmounting all other currently mounted file systems..."
        umount -a -d -r >/dev/null
        evaluate_retval
        ;;

    *)
        echo "Usage: ${0} {start|stop}"
        exit 1
        ;;
esac

# End $rc_base/init.d/mountfs
```

D.11. /etc/rc.d/init.d/udev_retry

```
#!/bin/sh
########################################################################
# Begin $rc_base/init.d/udev_retry
#
# Description : Udev cold-plugging script (retry)
#
# Authors     : Alexander E. Patrakov
#
# Version     : 00.02
#
# Notes       :
#
########################################################################

. /etc/sysconfig/rc
. ${rc_functions}

case "${1}" in
    start)
        boot_mesg "Retrying failed uevents, if any..."

        # From Debian: "copy the rules generated before / was mounted
        # read-write":
        for file in /dev/.udev/tmp-rules--*; do
            dest=${file##*tmp-rules--}
            [ "$dest" = '*' ] && break
            cat $file >> /etc/udev/rules.d/$dest
            rm -f $file
        done

        # Re-trigger the failed uevents in hope they will succeed now
        /sbin/udevadm trigger --type=failed --action=add

        # Now wait for udevd to process the uevents we triggered
        /sbin/udevadm settle
        evaluate_retval
        ;;

    *)
        echo "Usage ${0} {start}"
        exit 1
        ;;
esac

# End $rc_base/init.d/udev_retry
```

D.12. /etc/rc.d/init.d/cleanfs

```
#!/bin/sh
########################################################################
# Begin $rc_base/init.d/cleanfs
#
# Description : Clean file system
#
# Authors     : Gerard Beekmans - gerard@linuxfromscratch.org
#
# Version     : 00.00
#
```

```
# Notes       :
#
##############################################################################

. /etc/sysconfig/rc
. ${rc_functions}

# Function to create files/directory on boot.
create_files() {
    # Read in the configuration file.
    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
                                ;;
                            *)
                                boot_mesg -n "\nUnknown device type: ${dtype}" ${WARNING}
                                boot_mesg "" ${NORMAL}
                                ;;
                        esac
                        ;;
                    *)
                        boot_mesg -n "\nUnknown type: ${type}" ${WARNING}
                        boot_mesg "" ${NORMAL}
                        continue
                        ;;
                esac

                # Set up the permissions, too.
                chown ${usr}:${grp} "${name}"
                chmod ${perm} "${name}"
            fi
        done
    exec 0>&9 9>&-
}
```

```
case "${1}" in
    start)
        boot_mesg -n "Cleaning file systems:" ${INFO}

        boot_mesg -n " /tmp" ${NORMAL}
        cd /tmp &&
        find . -xdev -mindepth 1 ! -name lost+found \
            -delete || failed=1

        boot_mesg -n " /var/lock" ${NORMAL}
        cd /var/lock &&
        find . -type f -exec rm -f {} \; || failed=1

        boot_mesg " /var/run" ${NORMAL}
        cd /var/run &&
        find . ! -type d ! -name utmp \
            -exec rm -f {} \; || failed=1
        > /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
            boot_mesg "Creating files and directories..."
            create_files
            evaluate_retval
        fi
        ;;
    *)
        echo "Usage: ${0} {start}"
        exit 1
        ;;
esac

# End $rc_base/init.d/cleanfs
```

D.13. /etc/rc.d/init.d/console

```
#!/bin/sh
########################################################################
# Begin $rc_base/init.d/console
#
# Description : Sets keymap and screen font
#
# Authors     : Gerard Beekmans - gerard@linuxfromscratch.org
#        Alexander E. Patrakov
#
# Version     : 00.03
#
# Notes       :
#
########################################################################

. /etc/sysconfig/rc
. ${rc_functions}
```

```
# Native English speakers probably don't have /etc/sysconfig/console at all
if [ -f /etc/sysconfig/console ]
then
    . /etc/sysconfig/console
else
        exit 0
fi

is_true() {
    [ "$1" = "1" ] || [ "$1" = "yes" ] || [ "$1" = "true" ]
}

failed=0

case "${1}" in
    start)
        boot_mesg "Setting up Linux console..."
        # There should be no bogus failures below this line!

        # 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.
        # FIXME: Fedora Core also initializes two spare consoles
        # - do we want that?

        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
        is_true "${USE_FB}" || [ -z "${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
        ;;
    *)
        echo $"Usage:" "${0} {start}"
        exit 1
        ;;
esac

# End $rc_base/init.d/console
```

D.14. /etc/rc.d/init.d/localnet

```
#!/bin/sh
########################################################################
# Begin $rc_base/init.d/localnet
#
# Description : Loopback device
#
# Authors     : Gerard Beekmans  - gerard@linuxfromscratch.org
#
# Version     : 00.00
#
# Notes       :
#
########################################################################

. /etc/sysconfig/rc
. ${rc_functions}
. /etc/sysconfig/network

case "${1}" in
    start)
        boot_mesg "Bringing up the loopback interface..."
        ip addr add 127.0.0.1/8 label lo dev lo
        ip link set lo up
        evaluate_retval

        boot_mesg "Setting hostname to ${HOSTNAME}..."
        hostname ${HOSTNAME}
        evaluate_retval
        ;;

    stop)
        boot_mesg "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

# End $rc_base/init.d/localnet
```

D.15. /etc/rc.d/init.d/sysctl

```
#!/bin/sh
########################################################################
# Begin $rc_base/init.d/sysctl
#
# Description : File uses /etc/sysctl.conf to set kernel runtime
#               parameters
#
# Authors     : Nathan Coulson (nathan@linuxfromscratch.org)
#               Matthew Burgress (matthew@linuxfromscratch.org)
#
# Version     : 00.00
#
# Notes       :
#
########################################################################

. /etc/sysconfig/rc
. ${rc_functions}

case "${1}" in
    start)
        if [ -f "/etc/sysctl.conf" ]; then
            boot_mesg "Setting kernel runtime parameters..."
            sysctl -q -p
            evaluate_retval
        fi
        ;;

    status)
        sysctl -a
        ;;

    *)
        echo "Usage: ${0} {start|status}"
        exit 1
        ;;
esac

# End $rc_base/init.d/sysctl
```

D.16. /etc/rc.d/init.d/sysklogd

```
#!/bin/sh
```

```
###############################################################################
# Begin $rc_base/init.d/sysklogd
#
# Description : Sysklogd loader
#
# Authors     : Gerard Beekmans - gerard@linuxfromscratch.org
#
# Version     : 00.00
#
# Notes       :
#
###############################################################################

. /etc/sysconfig/rc
. ${rc_functions}

case "${1}" in
    start)
        boot_mesg "Starting system log daemon..."
        loadproc syslogd -m 0

        boot_mesg "Starting kernel log daemon..."
        loadproc klogd
        ;;

    stop)
        boot_mesg "Stopping kernel log daemon..."
        killproc klogd

        boot_mesg "Stopping system log daemon..."
        killproc syslogd
        ;;

    reload)
        boot_mesg "Reloading system log daemon config file..."
        reloadproc syslogd
        ;;

    restart)
        ${0} stop
        sleep 1
        ${0} start
        ;;

    status)
        statusproc syslogd
        statusproc klogd
        ;;

    *)
        echo "Usage: ${0} {start|stop|reload|restart|status}"
        exit 1
        ;;
esac

# End $rc_base/init.d/sysklogd
```

D.17. /etc/rc.d/init.d/network

```
#!/bin/sh
```

```
########################################################################
# Begin $rc_base/init.d/network
#
# Description : Network Control Script
#
# Authors     : Gerard Beekmans - gerard@linuxfromscratch.org
#         Nathan Coulson - nathan@linuxfromscratch.org
#         Kevin P. Fleming - kpfleming@linuxfromscratch.org
#
# Version     : 00.00
#
# Notes       :
#
########################################################################

. /etc/sysconfig/rc
. ${rc_functions}
. /etc/sysconfig/network

case "${1}" in
    start)
        # Start all network interfaces
        for file in ${network_devices}/ifconfig.*
        do
            interface=${file##*/ifconfig.}

            # skip if $file is * (because nothing was found)
            if [ "${interface}" = "*" ]
            then
                continue
            fi

            IN_BOOT=1 ${network_devices}/ifup ${interface}
        done
        ;;

    stop)
        # Reverse list
        FILES=""
        for file in ${network_devices}/ifconfig.*
        do
            FILES="${file} ${FILES}"
        done

        # Stop all network interfaces
        for file in ${FILES}
        do
            interface=${file##*/ifconfig.}

            # skip if $file is * (because nothing was found)
            if [ "${interface}" = "*" ]
            then
                continue
            fi

            IN_BOOT=1 ${network_devices}/ifdown ${interface}
        done
        ;;

    restart)
        ${0} stop
```

```
        sleep 1
        ${0} start
        ;;

    *)
        echo "Usage: ${0} {start|stop|restart}"
        exit 1
        ;;
esac

# End /etc/rc.d/init.d/network
```

D.18. /etc/rc.d/init.d/sendsignals

```
#!/bin/sh
########################################################################
# Begin $rc_base/init.d/sendsignals
#
# Description : Sendsignals Script
#
# Authors     : Gerard Beekmans - gerard@linuxfromscratch.org
#
# Version     : 00.00
#
# Notes       :
#
########################################################################

. /etc/sysconfig/rc
. ${rc_functions}

case "${1}" in
    stop)
        boot_mesg "Sending all processes the TERM signal..."
        killall5 -15
        error_value=${?}

        sleep ${KILLDELAY}

        if [ "${error_value}" = 0 -o "${error_value}" = 2 ]; then
            echo_ok
        else
            echo_failure
        fi

        boot_mesg "Sending all processes the KILL signal..."
        killall5 -9
        error_value=${?}

        sleep ${KILLDELAY}

        if [ "${error_value}" = 0 -o "${error_value}" = 2 ]; then
            echo_ok
        else
            echo_failure
        fi
        ;;

    *)
        echo "Usage: ${0} {stop}"
```

```
        exit 1
        ;;

esac

# End $rc_base/init.d/sendsignals
```

D.19. /etc/rc.d/init.d/reboot

```
#!/bin/sh
########################################################################
# Begin $rc_base/init.d/reboot
#
# Description : Reboot Scripts
#
# Authors     : Gerard Beekmans - gerard@linuxfromscratch.org
#
# Version     : 00.00
#
# Notes       :
#
########################################################################

. /etc/sysconfig/rc
. ${rc_functions}

case "${1}" in
    stop)
        boot_mesg "Restarting system..."
        reboot -d -f -i
        ;;

    *)
        echo "Usage: ${0} {stop}"
        exit 1
        ;;

esac

# End $rc_base/init.d/reboot
```

D.20. /etc/rc.d/init.d/halt

```
#!/bin/sh
########################################################################
# Begin $rc_base/init.d/halt
#
# Description : Halt Script
#
# Authors     : Gerard Beekmans - gerard@linuxfromscratch.org
#
# Version     : 00.00
#
# Notes       :
#
########################################################################

. /etc/sysconfig/rc
. ${rc_functions}
```

```
case "${1}" in
    stop)
        halt -d -f -i -p
        ;;
    *)
        echo "Usage: {stop}"
        exit 1
        ;;
esac

# End $rc_base/init.d/halt
```

D.21. /etc/rc.d/init.d/template

```
#!/bin/sh
########################################################################
# Begin $rc_base/init.d/
#
# Description :
#
# Authors     :
#
# Version     : 00.00
#
# Notes       :
#
########################################################################

. /etc/sysconfig/rc
. ${rc_functions}

case "${1}" in
    start)
        boot_mesg "Starting..."
        loadproc
        ;;

    stop)
        boot_mesg "Stopping..."
        killproc
        ;;

    reload)
        boot_mesg "Reloading..."
        reloadproc
        ;;

    restart)
        ${0} stop
        sleep 1
        ${0} start
        ;;

    status)
        statusproc
        ;;

    *)
        echo "Usage: ${0} {start|stop|reload|restart|status}"
        exit 1
```

```
        ;;
esac

# End $rc_base/init.d/
```

D.22. /etc/sysconfig/rc

```
################################################################################
# Begin /etc/sysconfig/rc
#
# Description : rc script configuration
#
# Authors     :
#
# Version     : 00.00
#
# Notes       :
#
################################################################################

rc_base=/etc/rc.d
rc_functions=${rc_base}/init.d/functions
network_devices=/etc/sysconfig/network-devices

# End /etc/sysconfig/rc
```

D.23. /etc/sysconfig/modules

```
################################################################################
# Begin /etc/sysconfig/modules
#
# Description : Module auto-loading configuration
#
# Authors     :
#
# Version     : 00.00
#
# Notes       : The syntax of this file is as follows:
#          <module> [<arg1> <arg2> ...]
#
# Each module should be on it's own line, and any options that you want
# passed to the module should follow it.  The line deliminator is either
# a space or a tab.
################################################################################

# End /etc/sysconfig/modules
```

D.24. /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.25. /etc/sysconfig/network-devices/ifup

```
#!/bin/sh
########################################################################
# Begin $network_devices/ifup
#
# Description : Interface Up
#
# Authors     : Nathan Coulson - nathan@linuxfromscratch.org
#               Kevin P. Fleming - kpfleming@linuxfromscratch.org
#
# Version     : 00.00
#
# Notes       : the IFCONFIG variable is passed to the scripts found
#               in the services directory, to indicate what file the
#               service should source to get environmental variables.
#
########################################################################

. /etc/sysconfig/rc
. ${rc_functions}

# Collect a list of configuration files for our interface
if [ -n "${2}" ]; then
    for file in ${@#$1} # All parameters except $1
  do
        FILES="${FILES} ${network_devices}/ifconfig.${1}/${file}"
    done
elif [ -d "${network_devices}/ifconfig.${1}" ]; then
    FILES=`echo ${network_devices}/ifconfig.${1}/*`
else
    FILES="${network_devices}/ifconfig.${1}"
fi

boot_mesg "Bringing up the ${1} interface..."
boot_mesg_flush

# Process each configruation file
for file in ${FILES}; do
    # skip backup files
    if [ "${file}" != "${file%""~""}" ]; then
```

```
        continue
    fi

    if [ ! -f "${file}" ]; then
        boot_mesg "${file} is not a network configuration file or directory." ${WARNING}
        echo_warning
        continue
    fi

    (
        . ${file}

        # Will not process this service if started by boot, and ONBOOT
        # is not set to yes
        if [ "${IN_BOOT}" = "1" -a "${ONBOOT}" != "yes" ]; then
            continue
        fi
        # Will not process this service if started by hotplug, and
        # ONHOTPLUG is not set to yes
        if [ "${IN_HOTPLUG}" = "1" -a "${ONHOTPLUG}" != "yes" \
                    -a "${HOSTNAME}" != "(none)" ]; then continue
        fi

        if [ -n "${SERVICE}" -a -x "${network_devices}/services/${SERVICE}" ]; then
            if [ -z "${CHECK_LINK}" -o "${CHECK_LINK}" = "y" \
                            -o "${CHECK_LINK}" = "yes" -o "${CHECK_LINK}" = "1" ]; then
                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
                    boot_mesg "Interface ${1} doesn't exist." ${WARNING}
                    echo_warning
                    continue
                fi
            fi
            IFCONFIG=${file} ${network_devices}/services/${SERVICE} ${1} up
        else
            boot_mesg "Unable to process ${file}.  Either" ${FAILURE}
            boot_mesg " the SERVICE variable was not set,"
            boot_mesg " or the specified service cannot be executed."
            echo_failure
            continue
        fi
    )
done

# End $network_devices/ifup
```

D.26. /etc/sysconfig/network-devices/ifdown

```
#!/bin/sh
########################################################################
# Begin $network_devices/ifdown
#
# Description : Interface Down
#
```

```
# Authors     : Nathan Coulson - nathan@linuxfromscratch.org
#               Kevin P. Fleming - kpfleming@linuxfromscratch.org
#
# Version     : 00.01
#
# Notes       : the IFCONFIG variable is passed to the scripts found
#               in the services directory, to indicate what file the
#               service should source to get environmental variables.
#
########################################################################

. /etc/sysconfig/rc
. ${rc_functions}

# Collect a list of configuration files for our interface
if [ -n "${2}" ]; then
    for file in ${@#$1}; do # All parameters except $1
        FILES="${FILES} ${network_devices}/ifconfig.${1}/${file}"
    done
elif [ -d "${network_devices}/ifconfig.${1}" ]; then
    FILES=`echo ${network_devices}/ifconfig.${1}/*`
else
    FILES="${network_devices}/ifconfig.${1}"
fi

# Reverse the order configuration files are processed in
for file in ${FILES}; do
    FILES2="${file} ${FILES2}"
done
FILES=${FILES2}

# Process each configuration file
for file in ${FILES}; do
    # skip backup files
    if [ "${file}" != "${file%""~""}" ]; then
        continue
    fi

    if [ ! -f "${file}" ]; then
        boot_mesg "${file} is not a network configuration file or directory." ${WARNING}
        echo_warning
        continue
    fi
    (
        . ${file}

        # Will not process this service if started by boot, and ONBOOT
        # is not set to yes
        if [ "${IN_BOOT}" = "1" -a "${ONBOOT}" != "yes" ]; then
            continue
        fi

        # Will not process this service if started by hotplug, and
        # ONHOTPLUG is not set to yes
        if [ "${IN_HOTPLUG}" = "1" -a "${ONHOTPLUG}" != "yes" ]; then
            continue
        fi

        # This will run the service script, if SERVICE is set
        if [ -n "${SERVICE}" -a -x "${network_devices}/services/${SERVICE}" ]; then
            if ip link show ${1} > /dev/null 2>&1
```

```
            then
                IFCONFIG=${file} ${network_devices}/services/${SERVICE} ${1} down
            else
                boot_mesg "Interface ${1} doesn't exist." ${WARNING}
                echo_warning
            fi
        else
            boot_mesg -n "Unable to process ${file}.  Either" ${FAILURE}
            boot_mesg -n " the SERVICE variable was not set,"
            boot_mesg " or the specified service cannot be executed."
            echo_failure
            continue
        fi
    )
done

if [ -z "${2}" ]; then
    link_status=`ip link show $1 2>/dev/null`
    if [ -n "${link_status}" ]; then
        if echo "${link_status}" | grep -q UP; then
            boot_mesg "Bringing down the ${1} interface..."
            ip link set ${1} down
            evaluate_retval
        fi
    fi
fi

# End $network_devices/ifdown
```

D.27. /etc/sysconfig/network-devices/services/ipv4-static

```
#!/bin/sh
########################################################################
# Begin $network_devices/services/ipv4-static
#
# Description : IPV4 Static Boot Script
#
# Authors     : Nathan Coulson - nathan@linuxfromscratch.org
#        Kevin P. Fleming - kpfleming@linuxfromscratch.org
#
# Version     : 00.00
#
# Notes       :
#
########################################################################

. /etc/sysconfig/rc
. ${rc_functions}
. ${IFCONFIG}

if [ -z "${IP}" ]; then
    boot_mesg "IP variable missing from ${IFCONFIG}, cannot continue." ${FAILURE}
    echo_failure
    exit 1
fi

if [ -z "${PREFIX}" -a -z "${PEER}" ]; then
    boot_mesg -n "PREFIX variable missing from ${IFCONFIG}," ${WARNING}
```

```
    boot_mesg " assuming 24."
    echo_warning
    PREFIX=24
    args="${args} ${IP}/${PREFIX}"
elif [ -n "${PREFIX}" -a -n "${PEER}" ]; then
    boot_mesg "PREFIX and PEER both specified in ${IFCONFIG}, cannot continue." ${FAILURE}
    echo_failure
    exit 1
elif [ -n "${PREFIX}" ]; then
    args="${args} ${IP}/${PREFIX}"
elif [ -n "${PEER}" ]; then
    args="${args} ${IP} peer ${PEER}"
fi

if [ -n "${BROADCAST}" ]; then
    args="${args} broadcast ${BROADCAST}"
fi

case "${2}" in
    up)
        boot_mesg "Adding IPv4 address ${IP} to the ${1} interface..."
        ip addr add ${args} dev ${1}
        evaluate_retval

        if [ -n "${GATEWAY}" ]; then
            if ip route | grep -q default; then
                boot_mesg "Gateway already setup; skipping." ${WARNING}
                echo_warning
            else
                boot_mesg "Setting up default gateway..."
                ip route add default via ${GATEWAY} dev ${1}
                evaluate_retval
             fi
        fi
    ;;

    down)
        if [ -n "${GATEWAY}" ];    then
            boot_mesg "Removing default gateway..."
            ip route del default
            evaluate_retval
        fi

        boot_mesg "Removing IPv4 address ${IP} from the ${1} interface..."
        ip addr del ${args} dev ${1}
        evaluate_retval
    ;;

    *)
        echo "Usage: ${0} [interface] {up|down}"
        exit 1
    ;;
esac

# End $network_devices/services/ipv4-static
```

D.28. /etc/sysconfig/network-devices/services/ipv4-static-route

```
#!/bin/sh
########################################################################
# Begin $network_devices/services/ipv4-static-route
#
# Description : IPV4 Static Route Script
#
# Authors     : Kevin P. Fleming - kpfleming@linuxfromscratch.org
#
# Version     : 00.00
#
# Notes       :
#
########################################################################

. /etc/sysconfig/rc
. ${rc_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 "
    ;;

    (*)
        boot_mesg "Unknown route type (${TYPE}) in ${IFCONFIG}, cannot continue." ${FAILURE}
        echo_failure
        exit 1
    ;;
esac

if [ -n "${need_ip}" ]; then
    if [ -z "${IP}" ]; then
        boot_mesg "IP variable missing from ${IFCONFIG}, cannot continue." ${FAILURE}
        echo_failure
        exit 1
    fi

    if [ -z "${PREFIX}" ]; then
        boot_mesg "PREFIX variable missing from ${IFCONFIG}, cannot continue." ${FAILURE}
```

```
        echo_failure
        exit 1
    fi

    args="${args} ${IP}/${PREFIX}"
    desc="${desc}${IP}/${PREFIX}"
fi

if [ -n "${need_gateway}" ]; then
    if [ -z "${GATEWAY}" ]; then
        boot_mesg "GATEWAY variable missing from ${IFCONFIG}, cannot continue." ${FAILURE}
        echo_failure
        exit 1
    fi
    args="${args} via ${GATEWAY}"
fi

if [ -n "${SOURCE}" ]; then
        args="${args} src ${SOURCE}"
fi

case "${2}" in
    up)
        boot_mesg "Adding '${desc}' route to the ${1} interface..."
        ip route add ${args} dev ${1}
        evaluate_retval
    ;;

    down)
        boot_mesg "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 $network_devices/services/ipv4-static-route
```

Приложение E. Udev configuration rules

The rules from udev-config-20100128.tar.bz2 in this appendix are listed for convenience. Installation is normally done via instructions in Раздел 6.60, «Udev-166».

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"
```

Приложение F. LFS Licenses

F.1. Creative Commons License

Creative Commons Legal Code

Attribution-NonCommercial-ShareAlike 2.0

Важно

CREATIVE COMMONS CORPORATION IS NOT A LAW FIRM AND DOES NOT PROVIDE LEGAL SERVICES. DISTRIBUTION OF THIS LICENSE DOES NOT CREATE AN ATTORNEY-CLIENT RELATIONSHIP. CREATIVE COMMONS PROVIDES THIS INFORMATION ON AN "AS-IS" BASIS. CREATIVE COMMONS MAKES NO WARRANTIES REGARDING THE INFORMATION PROVIDED, AND DISCLAIMS LIABILITY FOR DAMAGES RESULTING FROM ITS USE.

License

THE WORK (AS DEFINED BELOW) IS PROVIDED UNDER THE TERMS OF THIS CREATIVE COMMONS PUBLIC LICENSE ("CCPL" OR "LICENSE"). THE WORK IS PROTECTED BY COPYRIGHT AND/OR OTHER APPLICABLE LAW. ANY USE OF THE WORK OTHER THAN AS AUTHORIZED UNDER THIS LICENSE OR COPYRIGHT LAW IS PROHIBITED.

BY EXERCISING ANY RIGHTS TO THE WORK PROVIDED HERE, YOU ACCEPT AND AGREE TO BE BOUND BY THE TERMS OF THIS LICENSE. THE LICENSOR GRANTS YOU THE RIGHTS CONTAINED HERE IN CONSIDERATION OF YOUR ACCEPTANCE OF SUCH TERMS AND CONDITIONS.

1. Definitions
   a. "Collective Work" means a work, such as a periodical issue, anthology or encyclopedia, in which the Work in its entirety in unmodified form, along with a number of other contributions, constituting separate and independent works in themselves, are assembled into a collective whole. A work that constitutes a Collective Work will not be considered a Derivative Work (as defined below) for the purposes of this License.
   b. "Derivative Work" means a work based upon the Work or upon the Work and other pre-existing works, such as a translation, musical arrangement, dramatization, fictionalization, motion picture version, sound recording, art reproduction, abridgment, condensation, or any other form in which the Work may be recast, transformed, or adapted, except that a work that constitutes a Collective Work will not be considered a Derivative Work for the purpose of this License. For the avoidance of doubt, where the Work is a musical composition or sound recording, the synchronization of the Work in timed-relation with a moving image ("synching") will be considered a Derivative Work for the purpose of this License.
   c. "Licensor" means the individual or entity that offers the Work under the terms of this License.

d. "Original Author" means the individual or entity who created the Work.

e. "Work" means the copyrightable work of authorship offered under the terms of this License.

f. "You" means an individual or entity exercising rights under this License who has not previously violated the terms of this License with respect to the Work, or who has received express permission from the Licensor to exercise rights under this License despite a previous violation.

g. "License Elements" means the following high-level license attributes as selected by Licensor and indicated in the title of this License: Attribution, Noncommercial, ShareAlike.

2. Fair Use Rights. Nothing in this license is intended to reduce, limit, or restrict any rights arising from fair use, first sale or other limitations on the exclusive rights of the copyright owner under copyright law or other applicable laws.

3. License Grant. Subject to the terms and conditions of this License, Licensor hereby grants You a worldwide, royalty-free, non-exclusive, perpetual (for the duration of the applicable copyright) license to exercise the rights in the Work as stated below:

   a. to reproduce the Work, to incorporate the Work into one or more Collective Works, and to reproduce the Work as incorporated in the Collective Works;

   b. to create and reproduce Derivative Works;

   c. to distribute copies or phonorecords of, display publicly, perform publicly, and perform publicly by means of a digital audio transmission the Work including as incorporated in Collective Works;

   d. to distribute copies or phonorecords of, display publicly, perform publicly, and perform publicly by means of a digital audio transmission Derivative Works;

   The above rights may be exercised in all media and formats whether now known or hereafter devised. The above rights include the right to make such modifications as are technically necessary to exercise the rights in other media and formats. All rights not expressly granted by Licensor are hereby reserved, including but not limited to the rights set forth in Sections 4(e) and 4(f).

4. Restrictions.The license granted in Section 3 above is expressly made subject to and limited by the following restrictions:

   a. You may distribute, publicly display, publicly perform, or publicly digitally perform the Work only under the terms of this License, and You must include a copy of, or the Uniform Resource Identifier for, this License with every copy or phonorecord of the Work You distribute, publicly display, publicly perform, or publicly digitally perform. You may not offer or impose any terms on the Work that alter or restrict the terms of this License or the recipients' exercise of the rights granted hereunder. You may not sublicense the Work. You must keep intact all notices that refer to this License and to the disclaimer of warranties. You may not distribute, publicly display, publicly perform, or publicly digitally perform the Work with any technological measures that control access or use of the Work in a manner inconsistent with the terms of this License Agreement. The above applies to the Work as incorporated in a Collective Work, but this does not require the Collective Work apart from the Work itself to be made subject to the terms of this License. If You create a Collective Work, upon notice from any Licensor You must, to the extent practicable, remove from the Collective Work any reference to such Licensor or the Original Author, as requested. If You

create a Derivative Work, upon notice from any Licensor You must, to the extent practicable, remove from the Derivative Work any reference to such Licensor or the Original Author, as requested.

b. You may distribute, publicly display, publicly perform, or publicly digitally perform a Derivative Work only under the terms of this License, a later version of this License with the same License Elements as this License, or a Creative Commons iCommons license that contains the same License Elements as this License (e.g. Attribution-NonCommercial-ShareAlike 2.0 Japan). You must include a copy of, or the Uniform Resource Identifier for, this License or other license specified in the previous sentence with every copy or phonorecord of each Derivative Work You distribute, publicly display, publicly perform, or publicly digitally perform. You may not offer or impose any terms on the Derivative Works that alter or restrict the terms of this License or the recipients' exercise of the rights granted hereunder, and You must keep intact all notices that refer to this License and to the disclaimer of warranties. You may not distribute, publicly display, publicly perform, or publicly digitally perform the Derivative Work with any technological measures that control access or use of the Work in a manner inconsistent with the terms of this License Agreement. The above applies to the Derivative Work as incorporated in a Collective Work, but this does not require the Collective Work apart from the Derivative Work itself to be made subject to the terms of this License.

c. You may not exercise any of the rights granted to You in Section 3 above in any manner that is primarily intended for or directed toward commercial advantage or private monetary compensation. The exchange of the Work for other copyrighted works by means of digital file-sharing or otherwise shall not be considered to be intended for or directed toward commercial advantage or private monetary compensation, provided there is no payment of any monetary compensation in connection with the exchange of copyrighted works.

d. If you distribute, publicly display, publicly perform, or publicly digitally perform the Work or any Derivative Works or Collective Works, You must keep intact all copyright notices for the Work and give the Original Author credit reasonable to the medium or means You are utilizing by conveying the name (or pseudonym if applicable) of the Original Author if supplied; the title of the Work if supplied; to the extent reasonably practicable, the Uniform Resource Identifier, if any, that Licensor specifies to be associated with the Work, unless such URI does not refer to the copyright notice or licensing information for the Work; and in the case of a Derivative Work, a credit identifying the use of the Work in the Derivative Work (e.g., "French translation of the Work by Original Author," or "Screenplay based on original Work by Original Author"). Such credit may be implemented in any reasonable manner; provided, however, that in the case of a Derivative Work or Collective Work, at a minimum such credit will appear where any other comparable authorship credit appears and in a manner at least as prominent as such other comparable authorship credit.

e. For the avoidance of doubt, where the Work is a musical composition:

   i. Performance Royalties Under Blanket Licenses. Licensor reserves the exclusive right to collect, whether individually or via a performance rights society (e.g. ASCAP, BMI, SESAC), royalties for the public performance or public digital performance (e.g. webcast) of the Work if that performance is primarily intended for or directed toward commercial advantage or private monetary compensation.

ii. Mechanical Rights and Statutory Royalties. Licensor reserves the exclusive right to collect, whether individually or via a music rights agency or designated agent (e.g. Harry Fox Agency), royalties for any phonorecord You create from the Work ("cover version") and distribute, subject to the compulsory license created by 17 USC Section 115 of the US Copyright Act (or the equivalent in other jurisdictions), if Your distribution of such cover version is primarily intended for or directed toward commercial advantage or private monetary compensation. 6. Webcasting Rights and Statutory Royalties. For the avoidance of doubt, where the Work is a sound recording, Licensor reserves the exclusive right to collect, whether individually or via a performance-rights society (e.g. SoundExchange), royalties for the public digital performance (e.g. webcast) of the Work, subject to the compulsory license created by 17 USC Section 114 of the US Copyright Act (or the equivalent in other jurisdictions), if Your public digital performance is primarily intended for or directed toward commercial advantage or private monetary compensation.

f. Webcasting Rights and Statutory Royalties. For the avoidance of doubt, where the Work is a sound recording, Licensor reserves the exclusive right to collect, whether individually or via a performance-rights society (e.g. SoundExchange), royalties for the public digital performance (e.g. webcast) of the Work, subject to the compulsory license created by 17 USC Section 114 of the US Copyright Act (or the equivalent in other jurisdictions), if Your public digital performance is primarily intended for or directed toward commercial advantage or private monetary compensation.

5. Representations, Warranties and Disclaimer

   UNLESS OTHERWISE MUTUALLY AGREED TO BY THE PARTIES IN WRITING, LICENSOR OFFERS THE WORK AS-IS AND MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND CONCERNING THE WORK, EXPRESS, IMPLIED, STATUTORY OR OTHERWISE, INCLUDING, WITHOUT LIMITATION, WARRANTIES OF TITLE, MERCHANTIBILITY, FITNESS FOR A PARTICULAR PURPOSE, NONINFRINGEMENT, OR THE ABSENCE OF LATENT OR OTHER DEFECTS, ACCURACY, OR THE PRESENCE OF ABSENCE OF ERRORS, WHETHER OR NOT DISCOVERABLE. SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OF IMPLIED WARRANTIES, SO SUCH EXCLUSION MAY NOT APPLY TO YOU.

6. Limitation on Liability. EXCEPT TO THE EXTENT REQUIRED BY APPLICABLE LAW, IN NO EVENT WILL LICENSOR BE LIABLE TO YOU ON ANY LEGAL THEORY FOR ANY SPECIAL, INCIDENTAL, CONSEQUENTIAL, PUNITIVE OR EXEMPLARY DAMAGES ARISING OUT OF THIS LICENSE OR THE USE OF THE WORK, EVEN IF LICENSOR HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

7. Termination

   a. This License and the rights granted hereunder will terminate automatically upon any breach by You of the terms of this License. Individuals or entities who have received Derivative Works or Collective Works from You under this License, however, will not have their licenses terminated provided such individuals or entities remain in full compliance with those licenses. Sections 1, 2, 5, 6, 7, and 8 will survive any termination of this License.

   b. Subject to the above terms and conditions, the license granted here is perpetual (for the duration of the applicable copyright in the Work). Notwithstanding the above, Licensor reserves the right to release the Work under different license terms or to stop distributing the Work at any time; provided, however that any such election will

not serve to withdraw this License (or any other license that has been, or is required to be, granted under the terms of this License), and this License will continue in full force and effect unless terminated as stated above.

8. Miscellaneous

   a. Each time You distribute or publicly digitally perform the Work or a Collective Work, the Licensor offers to the recipient a license to the Work on the same terms and conditions as the license granted to You under this License.

   b. Each time You distribute or publicly digitally perform a Derivative Work, Licensor offers to the recipient a license to the original Work on the same terms and conditions as the license granted to You under this License.

   c. If any provision of this License is invalid or unenforceable under applicable law, it shall not affect the validity or enforceability of the remainder of the terms of this License, and without further action by the parties to this agreement, such provision shall be reformed to the minimum extent necessary to make such provision valid and enforceable.

   d. No term or provision of this License shall be deemed waived and no breach consented to unless such waiver or consent shall be in writing and signed by the party to be charged with such waiver or consent.

   e. This License constitutes the entire agreement between the parties with respect to the Work licensed here. There are no understandings, agreements or representations with respect to the Work not specified here. Licensor shall not be bound by any additional provisions that may appear in any communication from You. This License may not be modified without the mutual written agreement of the Licensor and You.

Важно

Creative Commons is not a party to this License, and makes no warranty whatsoever in connection with the Work. Creative Commons will not be liable to You or any party on any legal theory for any damages whatsoever, including without limitation any general, special, incidental or consequential damages arising in connection to this license. Notwithstanding the foregoing two (2) sentences, if Creative Commons has expressly identified itself as the Licensor hereunder, it shall have all rights and obligations of Licensor.

Except for the limited purpose of indicating to the public that the Work is licensed under the CCPL, neither party will use the trademark "Creative Commons" or any related trademark or logo of Creative Commons without the prior written consent of Creative Commons. Any permitted use will be in compliance with Creative Commons' then-current trademark usage guidelines, as may be published on its website or otherwise made available upon request from time to time.

Creative Commons may be contacted at *http://creativecommons.org/*.

F.2. The MIT License

Предметный указатель

Packages

Programs

Libraries

Scripts

Others