Linux From Scratch

Version 7.3

Created by Gerard Beekmans

Edited by Matthew Burgess and Bruce Dubbs

Copyright © 1999-2013 Gerard Beekmans

Copyright © 1999-2013, Gerard Beekmans

Table of Contents

7. Setting Up System Bootscripts

Index

Preface

Foreword

My journey to learn and better understand Linux began over a decade ago, back in 1998. I had just installed my first Linux distribution and had quickly become intrigued with the whole concept and philosophy behind Linux.

There are always many ways to accomplish a single task. The same can be said about Linux distributions. A great many have existed over the years. Some still exist, some have morphed into something else, yet others have been relegated to our memories. They all do things differently to suit the needs of their target audience. Because so many different ways to accomplish the same end goal exist, I began to realize I no longer had to be limited by any one implementation. Prior to discovering Linux, we simply put up with issues in other Operating Systems as you had no choice. It was what it was, whether you liked it or not. With Linux, the concept of choice began to emerge. If you didn't like something, you were free, even encouraged, to change it.

I tried a number of distributions and could not decide on any one. They were great systems in their own right. It wasn't a matter of right and wrong anymore. It had become a matter of personal taste. With all that choice available, it became apparent that there would not be a single system that would be perfect for me. So I set out to create my own Linux system that would fully conform to my personal preferences.

To truly make it my own system, I resolved to compile everything from source code instead of using pre-compiled binary packages. This "perfect" Linux system would have the strengths of various systems without their perceived weaknesses. At first, the idea was rather daunting. I remained committed to the idea that such a system could be built.

After sorting through issues such as circular dependencies and compile-time errors, I finally built a custom-built Linux system. It was fully operational and perfectly usable like any of the other Linux systems out there at the time. But it was my own creation. It was very satisfying to have put together such a system myself. The only thing better would have been to create each piece of software myself. This was the next best thing.

As I shared my goals and experiences with other members of the Linux community, it became apparent that there was a sustained interest in these ideas. It quickly became plain that such custom-built Linux systems serve not only to meet user specific requirements, but also serve as an ideal learning opportunity for programmers and system administrators to enhance their (existing) Linux skills. Out of this broadened interest, the *Linux From Scratch Project* was born.

This Linux From Scratch book is the central core around that project. It provides the background and instructions necessary for you to design and build your own system. While this book provides a template that will result in a correctly working system, you are free to alter the instructions to suit yourself, which is, in part, an important part of this project. You remain in control; we just lend a helping hand to get you started on your own journey.

I sincerely hope you will have a great time working on your own Linux From Scratch system and

enjoy the numerous benefits of having a system that is truly your own.

--
Gerard Beekmans
gerard AT linuxfromscratch D0T org

Audience

There are many reasons why you would want to read this book. One of the questions many people raise is, "why go through all the hassle of manually building a Linux system from scratch when you can just download and install an existing one?"

One important reason for this project's existence is to help you learn how a Linux system works from the inside out. Building an LFS system helps demonstrate what makes Linux tick, and how things work together and depend on each other. One of the best things that this learning experience can provide is the ability to customize a Linux system to suit your own unique needs.

Another key benefit of LFS is that it allows you to have more control over the system without relying on someone else's Linux implementation. With LFS, you are in the driver's seat and dictate every aspect of the system.

LFS allows you to create very compact Linux systems. When installing regular distributions, you are often forced to install a great many programs which are probably never used or understood. These programs waste resources. You may argue that with today's hard drive and CPUs, such resources are no longer a consideration. Sometimes, however, you are still constrained by size considerations if nothing else. Think about bootable CDs, USB sticks, and embedded systems. Those are areas where LFS can be beneficial.

Another advantage of a custom built Linux system is security. By compiling the entire system from source code, you are empowered to audit everything and apply all the security patches desired. It is no longer necessary to wait for somebody else to compile binary packages that fix a security hole. Unless you examine the patch and implement it yourself, you have no guarantee that the new binary package was built correctly and adequately fixes the problem.

The goal of Linux From Scratch is to build a complete and usable foundation-level system. If you do not wish to build your own Linux system from scratch, you may not entirely benefit from the information in this book.

There are too many other good reasons to build your own LFS system to list them all here. In the end, education is by far the most powerful of reasons. As you continue in your LFS experience, you will discover the power that information and knowledge truly bring.

LFS Target Architectures

The primary target architectures of LFS are the AMD/Intel x86 (32-bit) and x86_64 (64-bit) CPUs. On the other hand, the instructions in this book are also known to work, with some modifications, with the Power PC CPU. To build a system that utilizes one of these CPUs, the main prerequisite, in addition to those on the next few pages, is an existing Linux system such as an earlier LFS installation, Ubuntu, Red Hat/Fedora, SuSE, or other distribution that targets the architecture that you have. Also note that a 32-bit distribution can be installed and used as a host system on a 64-bit AMD/Intel computer.

Some other facts about 64-bit systems need to be added here. When compared to a 32-bit system, the sizes of executable programs are slightly larger and the execution speeds are only slightly faster. For example, in a test build of LFS-6.5 on a Core2Duo CPU based system, the following statistics were

measured:

```
Architecture Build Time     Build Size
32-bit       198.5 minutes  648 MB
64-bit       190.6 minutes  709 MB
```

As you can see, the 64-bit build is only 4% faster and is 9% larger than the 32-bit build. The gain from going to a 64-bit system is relatively minimal. Of course, if you have more than 4GB of RAM or want to manipulate data that exceeds 4GB, the advantages of a 64-bit system are substantial.

The default 64-bit build that results from LFS is considered a "pure" 64-bit system. That is, it supports 64-bit executables only. Building a "multi-lib" system requires compiling many applications twice, once for a 32-bit system and once for a 64-bit system. This is not directly supported in LFS because it would interfere with the educational objective of providing the instructions needed for a straightforward base Linux system. You can refer to the **Cross Linux From Scratch** project for this advanced topic.

There is one last comment about 64-bit systems. There are some older packages that cannot currently be built in a "pure" 64-bit system or require specialized build instructions. Generally, these packages have some embedded 32-bit specific assembly language instructions that fail when building on a 64-bit system. This includes some Xorg drivers for some legacy video cards at **http://xorg.freedesktop.org/releases/individual/driver/**. Many of these problems can be worked around, but may require some specialized procedures or patches.

LFS and Standards

The structure of LFS follows Linux standards as closely as possible. The primary standards are:

- **POSIX.1-2008**.

- **Filesystem Hierarchy Standard (FHS)**

- **Linux Standard Base (LSB) Specifications**

    The LSB has five separate standards: Core, C++, Desktop, Runtime Languages, and Printing. In addition to generic requirements there are also architecture specific requirements. LFS attempts to conform to the architectures discussed in the previous section.

    ### Note

    Many people do not agree with the requirements of the LSB. The main purpose of defining it is to ensure that proprietary software will be able to be installed and run properly on a compliant system. Since LFS is source based, the user has complete control over what packages are desired and many choose not to install some packages that are specified by the LSB.

Creating a complete LFS system capable of passing the LSB certifications tests is possible, but not without many additional packages that are beyond the scope of LFS. These additional packages have installation instructions in BLFS.

Packages supplied by LFS needed to satisfy the LSB Requirements

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:	None
LSB Runtime Languages:	Perl
LSB Printing:	None
LSB Multimeda:	None

Packages supplied by BLFS needed to satisfy the LSB Requirements

LSB Core:	At, Batch (a part of At), Bc, Cpio, Ed, Fcrontab, Initd-tools, Lsb_release, PAM, Sendmail (or Postfix or Exim)
LSB C++:	None
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, Xdg-utils

Packages not supplied by LFS or BLFS needed to satisfy the LSB Requirements

LSB Core:	None
LSB C++:	None
LSB Desktop:	None
LSB Runtime Languages:	None
LSB Printing:	None
LSB Multimeda:	None

Rationale for Packages in the Book

As stated earlier, the goal of LFS is to build a complete and usable foundation-level system. This includes all packages needed to replicate itself while providing a relatively minimal base from which to customize a more complete system based on the choices of the user. This does not mean that LFS is the smallest system possible. Several important packages are included that are not strictly required. The lists below document the rationale for each package in the book.

- Autoconf

  This package contains programs for producing shell scripts that can automatically configure source code from a developer's template. It is often needed to rebuild a package after updates to the build procedures.

- Automake

This package contains programs for generating Make files from a template. It is often needed to rebuild a package after updates to the build procedures.

- Bash

  This package satisfies an LSB core requirement to provide a Bourne Shell interface to the system. It was chosen over other shell packages because of its common usage and extensive capabilities beyond basic shell functions.

- Binutils

  This package contains a linker, an assembler, and other tools for handling object files. The programs in this package are needed to compile most of the packages in an LFS system and beyond.

- Bison

  This package contains the GNU version of yacc (Yet Another Compiler Compiler) needed to build several other LFS programs.

- Bzip2

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

- Check

  This package contains a test harness for other programs. It is only installed in the temporary toolchain.

- Coreutils

  This package contains a number of essential programs for viewing and manipulating files and directories. These programs are needed for command line file management, and are necessary for the installation procedures of every package in LFS.

- DejaGNU

  This package contains a framework for testing other programs. It is only installed in the temporary toolchain.

- Diffutils

  This package contains programs that show the differences between files or directories. These programs can be used to create patches, and are also used in many packages' build procedures.

- E2fsprogs

  This package contains the utilities for handling the ext2, ext3 and ext4 file systems. These are the most common and thoroughly tested file systems that Linux supports.

- Expect

  This package contains a program for carrying out scripted dialogues with other interactive programs. It is commonly used for testing other packages. It is only installed in the temporary toolchain.

- File

  This package contains a utility for determining the type of a given file or files. A few packages need it to build.

- Findutils

  This package contains programs to find files in a file system. It is used in many packages' build scripts.

- Flex

  This package contains a utility for generating programs that recognize patterns in text. It is the GNU version of the lex (lexical analyzer) program. It is required to build several LFS packages.

- Gawk

  This package contains programs for manipulating text files. It is the GNU version of awk (Aho-Weinberg-Kernighan). It is used in many other packages' build scripts.

- Gcc

  This package is the Gnu Compiler Collection. It contains the C and C++ compilers as well as several others not built by LFS.

- GDBM

  This package contains the GNU Database Manager library. It is used by one other LFS package, Man-DB.

- Gettext

  This package contains utilities and libraries for internationalization and localization of numerous packages.

- Glibc

  This package contains the main C library. Linux programs would not run without it.

- GMP

  This package contains math libraries that provide useful 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.

- Kmod

  This package contains programs needed to administer Linux kernel modules.

- 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.

- Libpipeline

  The Libpipeline package contains a library for manipulating pipelines of subprocesses in a flexible and convenient way. It is required by the Man-DB package.

- 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.

- 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 provides a program to return meta-data about an installed library or package.

- Procps-NG

  This package contains programs for monitoring processes. These programs are useful for system administration, and are also used by the LFS Bootscripts.

- Psmisc

  This package contains programs for displaying information about running processes. These programs are useful for system administration.

- Readline

  This package is a set of libraries that offers command-line editing and history capabilities. It is used by Bash.

- Sed

  This package allows editing of text without opening it in a text editor. It is also needed by most LFS packages' configure scripts.

- Shadow

  This package contains programs for handling passwords in a secure way.

- Sysklogd

  This package contains programs for logging system messages, such as those given by the kernel or daemon processes when unusual events occur.

- Sysvinit

  This package provides the init program, which is the parent of all other processes on the Linux system.

- Tar

  This package provides archiving and extraction capabilities of virtually all packages used in LFS.

- Tcl

  This package contains the Tool Command Language used in many test suites in LFS packages. It is only installed in the temporary toolchain.

- Texinfo

  This package contains programs for reading, writing, and converting info pages. It is used in the installation procedures of many LFS packages.

- Udev

  This package contains programs for dynamic creation of device nodes. It is an alternative to creating thousands of static devices in the /dev directory.

- Util-linux

  This package contains miscellaneous utility programs. Among them are utilities for handling file systems, consoles, partitions, and messages.

- Vim

  This package contains an editor. It was chosen because of its compatibility with the classic vi editor and its huge number of powerful capabilities. An editor is a very personal choice for many users and any other editor could be substituted if desired.

- XZ Utils

  This package contains programs for compressing and decompressing files. It provides the highest compression generally available and is useful for decompressing packages XZ or LZMA format.

- Zlib

  This package contains compression and decompression routines used by some programs.

Prerequisites

Building an LFS system is not a simple task. It requires a certain level of existing knowledge of Unix system administration in order to resolve problems and correctly execute the commands listed. In particular, as an absolute minimum, you should already have the ability to use the command line (shell) to copy or move files and directories, list directory and file contents, and change the current directory. It is also expected that you have a reasonable knowledge of using and installing Linux software.

Because the LFS book assumes *at least* this basic level of skill, the various LFS support forums are unlikely to be able to provide you with much assistance in these areas. You will find that your questions regarding such basic knowledge will likely go unanswered or you will simply be referred to the LFS essential pre-reading list.

Before building an LFS system, we recommend reading the following HOWTOs:

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

  This is a comprehensive guide to building and installing "generic" Unix software packages under Linux. Although it was written some time ago, it still provides a good summary of the basic techniques needed to build and install software.

- The Linux Users' Guide **http://tldp.org/pub/Linux/docs/ldp-archived/users-guide/**

This guide covers the usage of assorted Linux software. This reference is also fairly old, but still valid.

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

  This is an LFS Hint written specifically for users new to Linux. It includes a list of links to excellent sources of information on a wide range of topics. Anyone attempting to install LFS should have an understanding of many of the topics in this hint.

Host System Requirements

Your host system should have the following software with the minimum versions indicated. This should not be an issue for most modern Linux distributions. Also note that many distributions will place software headers into separate packages, often in the form of "<package-name>-devel" or "<package-name>-dev". Be sure to install those if your distribution provides them.

Earlier versions of the listed software packages may work, but has not been tested.

Bash-3.2 (/bin/sh should be a symbolic or hard link to bash)

Binutils-2.17 (Versions greater than 2.23.1 are not recommended as they have not been tested)

Bison-2.3 (/usr/bin/yacc should be a link to bison or small script that executes bison)

Bzip2-1.0.4

Coreutils-6.9

Diffutils-2.8.1

Findutils-4.2.31

Gawk-3.1.5 (/usr/bin/awk should be a link to gawk)

Gcc-4.1.2 (Versions greater than 4.7.2 are not recommended as they have not been tested)

Glibc-2.5.1 (Versions greater than 2.17 are not recommended as they have not been tested)

Grep-2.5.1a

Gzip-1.3.12

Linux Kernel-2.6.25 (having been compiled with GCC-4.1.2 or greater)

The reason for the kernel version requirement is that we specify that version when building glibc in Chapter 6 at the recommendation of the developers. It is also required by udev.

If the host kernel is either earlier than 2.6.25, or it was not compiled using a GCC-4.1.2 (or later) compiler, you will need to replace the kernel with one adhering to the specifications. There are two ways you can go about this. First, see if your Linux vendor provides a 2.6.25 or later kernel package. If so, you may wish to install it. If your vendor doesn't offer an acceptable kernel package, or you would prefer not to install it, you can compile a kernel yourself. Instructions for compiling the kernel and configuring the boot loader (assuming the host uses GRUB) are located in 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

Xz-5.0.0

Note that the symlinks mentioned above are required to build an LFS system using the instructions contained within this book. Symlinks that point to other software (such as dash, mawk, etc.) may work, but are not tested or supported by the LFS development team, and may require either deviation from the instructions or additional patches to some packages.

To see whether your host system has all the appropriate versions, and the ability to compile programs, run the following:

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

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

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

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

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

```
bash version-check.sh
```

Typography

To make things easier to follow, there are a few typographical conventions used throughout this book. This section contains some examples of the typographical format found throughout Linux From Scratch.

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

This form of text is designed to be typed exactly as seen unless otherwise noted in the surrounding text. It is also used in the explanation sections to identify which of the commands is being referenced.

In some cases, a logical line is extended to two or more physical lines with a backslash at the end of the line.

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

Note that the backslash must be followed by an immediate return. Other whitespace characters like spaces or tab characters will create incorrect results.

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

This form of text (fixed-width text) shows screen output, usually as the result of commands issued. This format is also used to show filenames, such as /etc/ld.so.conf.

Emphasis

This form of text is used for several purposes in the book. Its main purpose is to emphasize important points or items.

http://www.linuxfromscratch.org/

This format is used for hyperlinks both within the LFS community and to external pages. It includes HOWTOs, download locations, and websites.

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

This format is used when creating configuration files. The first command tells the system to create the file $LFS/etc/group from whatever is typed on the following lines until the sequence End Of File (EOF) is encountered. Therefore, this entire section is generally typed as seen.

<REPLACED TEXT>

This format is used to encapsulate text that is not to be typed as seen or for copy-and-paste operations.

[OPTIONAL TEXT]

This format is used to encapsulate text that is optional.

```
passwd(5)
```

This format is used to refer to a specific manual (man) page. The number inside parentheses indicates a specific section inside the manuals. For example, **passwd** has two man pages. Per LFS installation instructions, those two man pages will be located at /usr/share/man/man1/passwd.1 and /usr/share/man/man5/passwd.5. When the book uses passwd(5) it is specifically referring to /usr/share/man/man5/passwd.5. **man passwd** will print the first man page it finds that matches "passwd", which will be /usr/share/man/man1/passwd.1. For this example, you will need to run **man 5 passwd** in order to read the specific page being referred to. It should be noted that most man pages do not have duplicate page names in different sections. Therefore, **man <program name>** is generally sufficient.

Structure

This book is divided into the following parts.

Part I - Introduction

Part I explains a few important notes on how to proceed with the LFS installation. This section also provides meta-information about the book.

Part II - Preparing for the Build

Part II describes how to prepare for the building process—making a partition, downloading the packages, and compiling temporary tools.

Part III - Building the LFS System

Part III guides the reader through the building of the LFS system—compiling and installing all the packages one by one, setting up the boot scripts, and installing the kernel. The resulting Linux system is the foundation on which other software can be built to expand the system as desired. At the end of this book, there is an easy to use reference listing all of the programs, libraries, and important files that have been installed.

Errata

The software used to create an LFS system is constantly being updated and enhanced. Security warnings and bug fixes may become available after the LFS book has been released. To check whether the package versions or instructions in this release of LFS need any modifications to accommodate security vulnerabilities or other bug fixes, please visit **http://www.linuxfromscratch.org/lfs/errata/7.3/** before proceeding with your build. You should note any changes shown and apply them to the relevant section of the book as you progress with building the LFS system.

Part I. Introduction

Chapter 1. Introduction

1.1. How to Build an LFS System

The LFS system will be built by using an already installed Linux distribution (such as Debian, Mandriva, Red Hat, or SUSE). This existing Linux system (the host) will be used as a starting point to provide necessary programs, including a compiler, linker, and shell, to build the new system. Select the "development" option during the distribution installation to be able to access these tools.

As an alternative to installing a separate distribution onto your machine, you may wish to use a LiveCD from a commercial distribution.

Chapter 2 of this book describes how to create a new Linux native partition and file system. This is the place where the new LFS system will be compiled and installed. Chapter 3 explains which packages and patches need to be downloaded to build an LFS system and how to store them on the new file system. Chapter 4 discusses the setup of an appropriate working environment. Please read Chapter 4 carefully as it explains several important issues you need be aware of before beginning to work your way through Chapter 5 and beyond.

Chapter 5 explains the installation of a number of packages that will form the basic development suite (or toolchain) which is used to build the actual system in Chapter 6. Some of these packages are needed to resolve circular dependencies—for example, to compile a compiler, you need a compiler.

Chapter 5 also shows you how to build a first pass of the toolchain, including Binutils and GCC (first pass basically means these two core packages will be reinstalled). The next step is to build Glibc, the C library. Glibc will be compiled by the toolchain programs built in the first pass. Then, a second pass of the toolchain will be built. This time, the toolchain will be dynamically linked against the newly built Glibc. The remaining Chapter 5 packages are built using this second pass toolchain. When this is done, the LFS installation process will no longer depend on the host distribution, with the exception of the running kernel.

This effort to isolate the new system from the host distribution may seem excessive. A full technical explanation as to why this is done is provided in Section 5.2, "Toolchain Technical Notes".

In Chapter 6, the full LFS system is built. The `chroot` (change root) program is used to enter a virtual environment and start a new shell whose root directory will be set to the LFS partition. This is very similar to rebooting and instructing the kernel to mount the LFS partition as the root partition. The system does not actually reboot, but instead `chroot`'s because creating a bootable system requires additional work which is not necessary just yet. The major advantage is that "chrooting" allows you to continue using the host system while LFS is being built. While waiting for package compilations to complete, you can continue using your computer as normal.

To finish the installation, the LFS-Bootscripts are set up in Chapter 7, and the kernel and boot loader are set up in Chapter 8. Chapter 9 contains information on continuing the LFS experience beyond this book. After the steps in this book have been implemented, the computer will be ready to reboot into the new LFS system.

This is the process in a nutshell. Detailed information on each step is discussed in the following chapters and package descriptions. Items that may seem complicated will be clarified, and everything will fall into place as you embark on the LFS adventure.

1.2. What's new since the last release

Below is a list of package updates made since the previous release of the book.

Upgraded to:

- Automake 1.13.1

- Binutils 2.23.1

- Bison 2.7

- Check 0.9.9

- Coreutils 8.21

- E2fsprogs 1.42.7

- Gawk 4.0.2

- GCC 4.7.2

- Gettext 0.18.2

- Glibc 2.17

- GMP 5.1.1

- Groff 1.22.2

- IPRoute2 3.8.0

- Kbd 1.15.5

- Kmod 12

- Less 451

- Libpipeline 1.2.2

- Linux 3.8.1

- Man-DB 2.6.3

- Man-pages 3.47

- MPC 1.0.1

- Patch 2.7.1

- Pkg-config 0.28

- Procps-ng 3.3.6

- Psmisc 22.20

- Sed 4.2.2

- TCL 8.6.0

- Texinfo 5.0

- Tzdata 2012j

- Udev 197 (extracted from systemd-197)

- Util-Linux 2.22.2

Added:

- bash-4.2-fixes-11.patch

- binutils-2.23.1-testsuite_fix-1.patch

- coreutils-8.21-i18n-1.patch

- kbd-1.15.5-backspace-1.patch

- make-3.82-upstream_fixes-3.patch

Removed:

- bash-4.2-fixes-8.patch

- binutils-2.22-build_fix-1.patch

- coreutils-8.19-i18n-1.patch

- gcc-4.7.1-startfiles_fix-1.patch

- glibc-2.16.0-fix_test_installation-1.patch

- glibc-2.16.0-res_query_fix-1.patch

- kbd-1.15.3-backspace-1.patch

- kbd-1.15.3-upstream_fixes-1.patch

- kmod-9-testsuite-1.patch

- make-3.82-upstream_fixes-2.patch

- patch-2.6.1-test_fix-1.patch

- procps-3.2.8-fix_HZ_errors-1.patch

- procps-3.2.8-watch_unicode-1.patch

- sed-4.2.1-testsuite_fixes-1.patch

1.3. Changelog

This is version 7.3 of the Linux From Scratch book, dated March 1, 2013. If this book is more than six months old, a newer and better version is probably already available. To find out, please check one of the mirrors via **http://www.linuxfromscratch.org/mirrors.html**.

Below is a list of changes made since the previous release of the book.

Changelog Entries:

- 2013-03-01
  - [bdubbs] - Upgrade to Linux-3.8.1. Fixes **#3295**.

- 2013-02-27
  - [bdubbs] - Fix a potential test error in procps-ng. Thanks to Pierre Labastie for the patch. Fixes **#3293**.

- 2013-02-26
  - [bdubbs] - Upgrade to File-5.13. Fixes **#3292**.

- 2013-02-26
  - [ken] - Remove the obsolete resizecons program from kbd, again.

- 2013-02-24
  - [matthew] - Upgrade to IPRoute2-3.8.0. Fixes **#3291**.
  - [matthew] - Upgrade to Linux-3.8. Fixes **#3290**.

- 2013-02-19
  - [bdubbs] - Update file name for adjusting CD-ROM rules mode in section 7.5.1.

- 2013-02-18
  - [bdubbs] - Upgrade to Coreutils-8.21. Fixes **#3286**.
  - [bdubbs] - Upgrade to Texinfo-5.0. Fixes **#3284**.
  - [bdubbs] - Upgrade to Linux-3.7.9. Fixes **#3281**.

- 2013-02-13
  - [matthew] - Upgrade to Man-pages-3.47. Fixes **#3284**.
  - [matthew] - Upgrade to GMP-5.1.1. Fixes **#3283**.
  - [matthew] - Use latest Coreutils-i18n patch from Fedora. Fixes **#3282**.
  - [matthew] - Upgrade to Linux-3.7.7. Fixes **#3281**.
  - [matthew] - Upgrade to Groff-1.22.2. Fixes **#3280**.

- 2013-01-30
  - [bdubbs] - Change ncurses instructions to create and install .pc files.

- 2013-01-29
  - [matthew] - Upgrade to Man-Pages-3.46. Fixes **#3278**.
  - [matthew] - Upgrade to Linux-3.7.5. Fixes **#3277**.

- 2013-01-27
  - [bdubbs] - Update to pkg-config-0.28. Fixes **#3276**

- 2013-01-25
  - [bdubbs] - Revise procps-ng install instructions to place files in the proper locations.

- 2013-01-24
  - [bdubbs] - Remove bashisms from init-functions file in the boot scripts.

- 2013-01-24
  - [bdubbs] - Upgrade to e2fsprogs-1.42.7. Fixes **#3274**.

- 2013-01-22

- ○ [bdubbs] - Upgrade to Linux-3.7.4. Fixes **#3273**.

- 2013-01-21
  - ○ [bdubbs] - Upgrade to Procps-ng-3.3.6. Fixes **#3095**.

- 2013-01-20
  - ○ [matthew] - Upgrade to Linux-3.7.3. Fixes **#3272**.

- 2013-01-11
  - ○ [bdubbs] - Revised explanation for /etc/modprobe.conf. Fixes **#3270**.
  - ○ [bdubbs] - Update udev-lfs init-net-rules.sh script for "en*" devices introduced in systemd-197.

- 2013-01-09
  - ○ [bdubbs] - Reformat 'Rebooting the System' recommendations.
  - ○ [bdubbs] - Update udev-lfs scripts. Update to systemd-197.

- 2013-01-02
  - ○ [matthew] - Upgrade to Bash-4.2.42. Fixes **#3268**.
  - ○ [matthew] - Upgrade to Groff-1.22.1. Fixes **#3266**.
  - ○ [matthew] - Upgrade to Automake-1.13.1. Fixes **#3265**.
  - ○ [matthew] - Upgrade to Coreutils-8.20. Fixes **#3215**.

- 2012-12-31
  - ○ [bdubbs] - Add patch to fix binutils test suite. All binutils tests now pass, so remove the "-k" flag from test invocation. Thanks to Pierre Labastie for the patch.

- 2012-12-30
  - ○ [matthew] - Upgrade to Kbd-1.15.5. Fixes **#3239**.
  - ○ [matthew] - All E2fsprogs tests pass now, so remove the "-k" flag from its invocation.

- 2012-12-28
  - ○ [bdubbs] - Put traceroute in /bin for consistency. Fixes **#3264**.
  - ○ [bdubbs] - Fix the location for mounting /dev/shm inside chroot. Fixes **#3258**.
  - ○ [matthew] - Move the build of Procps to before E2fsprogs as the latter requires `ps` to be available during its testsuite run.
  - ○ [matthew] - Upgrade to Gettext-0.18.2. Fixes **#3263**.
  - ○ [matthew] - Upgrade to Gawk-4.0.2. Fixes **#3262**.
  - ○ [matthew] - Upgrade to Glibc-2.17. Fixes **#3261**.
  - ○ [matthew] - Upgrade to Sed-4.2.2. Fixes **#3260**.
  - ○ [matthew] - Upgrade to GMP-5.1.0. Fixes **#3259**.
  - ○ [matthew] - Upgrade to Tcl-8.6.0. Fixes **#3257**.
  - ○ [matthew] - Upgrade to Man-Pages-3.45. Fixes **#3256**.

- 2012-12-18
  - ○ [bdubbs] - Upgrade to Automake-1.12.6. Fixes **#3253**.
  - ○ [bdubbs] - Upgrade to Linux-3.7.1. Fixes **#3254**.

- 2012-12-16
  - [matthew] - Upgrade to Util-Linux-2.22.2. Fixes **#3250**.
  - [matthew] - Remove a few entries from the acronym list as they are no longer referenced in the book. Fixes **#3249**. Thanks to Chris Staub for the patch.
  - [matthew] - Upgrade to Bison-2.7. Fixes **#3247**.

- 2012-12-12
  - [matthew] - Upgrade to IPRoute2-3.7.0. Fixes **#3246**.
  - [matthew] - Update Check's list of installed programs. Fixes **#3245**. Thanks to Chris Staub for the patch.
  - [matthew] - All of Flex's tests pass, so remove the "-k" option to its testsuite invocation. Fixes **#3244**.
  - [matthew] - Upgrade to E2fsprogs-1.42.6. Fixes **#3243**.
  - [matthew] - Remove redundant --enable-addons parameter to Glibc's configure script. Fixes **#3241**.
  - [matthew] - Update IRC server information. Fixes **#3240**. Thanks to Chris Staub for the patch.
  - [matthew] - Upgrade to Kmod-12. Fixes **#3238**.
  - [matthew] - Upgrade to Linux-3.7. Fixes **#3237**.

- 2012-12-03
  - [bdubbs] - Fix build issues in makefile for systemd-196/udev-lfs-196.

- 2012-11-28
  - [bdubbs] - Update makefile and instructions for systemd-196/udev-lfs-196.

- 2012-11-27
  - [matthew] - Upgrade to Linux-3.6.8. Fixes **#3234**.

- 2012-11-26
  - [bdubbs] - Re-emphasize host system requirements in Chapter 5 General Compilation Instructions.

- 2012-11-22
  - [bdubbs] - Upgrade to systemd-196/udev-lfs-196. Fixes **#3233**.

- 2012-11-18
  - [matthew] - Upgrade to Linux-3.6.7. Fixes **#3232**.
  - [matthew] - Upgrade to Automake-1.12.5. Fixes **#3231**.

- 2012-11-14
  - [matthew] - Upgrade to Tzdata-2012j. Fixes **#3227**.
  - [matthew] - Upgrade to Binutils-2.23.1. Fixes **#3226**.

- 2012-11-13
  - [matthew] - Upgrade to Tcl-8.5.13. Fixes **#3224**.
  - [matthew] - Upgrade to Kmod-11. Fixes **#3223**.
  - [matthew] - Upgrade to Man-Pages-3.44. Fixes **#3222**.

- ○ [matthew] - Upgrade to Bison-2.6.5. Fixes **#3221**.

  ○ [matthew] - Upgrade to Tzdata-2012i. Fixes **#3220**.

  ○ [matthew] - Upgrade to Linux-3.6.6. Fixes **#3219**.

- 2012-11-03
  ○ [matthew] - Upgrade to Perl-5.16.2. Fixes **#3218**.

  ○ [matthew] - Upgrade to Bash-4.2.39. Fixes **#3217**.

- 2012-11-02
  ○ [matthew] - Change Freshmeat.net references to Freecode. Thanks to Chris Staub for the report and patch. Fixes **#3216**.

  ○ [matthew] - Upgrade to Check-0.9.9. Fixes **#3214**.

  ○ [matthew] - Upgrade to Bison-2.6.4. Fixes **#3212**.

  ○ [matthew] - Upgrade to Linux-3.6.5. Fixes **#3211**.

  ○ [matthew] - Upgrade to Tzdata-2012h. Fixes **#3209**.

  ○ [matthew] - Upgrade to Man-Pages-3.43. Fixes **#3208**.

  ○ [matthew] - Clean up a couple of instructions for Flex, made possible by the upgrade to Flex-2.5.37. Fixes **3206** and **3210**.

- 2012-11-01
  ○ [bdubbs] - Upgrade to systemd/lfs-udev-195. Fixes **#3197**.

- 2012-10-15
  ○ [bdubbs] - Add notes to the gcc and binutils sections in Chapter 6 about "link time optimization" and the extra files built by gcc. Fixes **#3200**.

  ○ [bdubbs] - Upgrade to tzdata-2012f. Fixes **#3205**.

  ○ [bdubbs] - Update installed program description for several packages. Thanks to Chris Staub for the patch. Fixes **#3203**.

- 2012-10-14
  ○ [bdubbs] - Remove utmpdump from sysvinit because it is now installed by util-linux. Thanks to Chris Staub for the patch. Fixes **#3202**.

  ○ [bdubbs] - Change procps to not install the kill program that is now installed by util-linux. Fixes **#3201**.

  ○ [bdubbs] - Update to util-linux-2.22.1. Fixes **#3199**.

  ○ [bdubbs] - Update to linux-3.6.2. Fixes **#3198**.

  ○ [bdubbs] - Add boot/shutdown script customization instructions.

- 2012-10-02
  ○ [matthew] - Upgrade to IPRoute2-3.6.0. Fixes **#3196**.

  ○ [matthew] - Upgrade to Linux-3.6. Fixes **#3195**.

  ○ [matthew] - Upgrade to Psmisc-22.20. Fixes **#3194**.

  ○ [matthew] - Upgrade to Patch-2.7.1. Fixes **#3193**.

  ○ [matthew] - Add a patch to allow Kmod's testsuite to pass on 32-bit systems. Fixes **#3191**.

  ○ [matthew] - Upgrade to GCC-4.7.2. Fixes **#3190**.

  ○ [matthew] - Use latest upstream fixes patch for Make, which enables WebKitGtk to build

with parallel builds. Fixes **#3188**.

- ○ [matthew] - Upgrade to Man-DB-2.6.3. Fixes **#3187**.

- ○ [matthew] - Upgrade to Libpipeline-1.2.2. Fixes **#3186**.

- ○ [matthew] - Upgrade to Automake-1.12.4. Fixes **#3185**.

- ○ [matthew] - Fix Flex instructions; the directory creation for its documentation is performed by the Makefile since r9999.

- • 2012-09-27

  - ○ [bdubbs] - Update to systemd/udev-lfs-193. Fixes **#3192**.

- • 2012-09-27

  - ○ [bdubbs] - Update to systemd/udev-lfs-192. Fixes **#3189**.

- • 2012-09-16

  - ○ [matthew] - Upgrade to Patch-2.7. Fixes **#3182**.

  - ○ [matthew] - Upgrade to MPC-1.0.1. Fixes **#3181**.

  - ○ [matthew] - Upgrade to Kmod-10. Fixes **#3180**.

  - ○ [matthew] - Fix link to Linux User's Guide. Fixes **#3179**.

  - ○ [matthew] - Upgrade to Less-451. Fixes **#3178**.

  - ○ [matthew] - Upgrade to Bash-4.2.37. Fixes **#3177**.

  - ○ [matthew] - Upgrade to Pkg-Config-0.27.1. Fixes **#3174**.

  - ○ [matthew] - Upgrade to Linux-3.5.4. Fixes **#3173**.

- • 2012-09-05

  - ○ [bdubbs] - Add udevadm trigger --action=change line to udev script to support initramfs better.

- • 2012-09-04

  - ○ [bdubbs] - Update grub packages and naming conventions example.

  - ○ [bdubbs] - Update to util-linux-2.22. Fixes **#3145**.

- • 2012-09-02

  - ○ [bdubbs] - Add a patch to Chapter 6 glibc to fix the test-installation.pl script instead of just preventing it from running. Fixes **#3175**.

  - ○ [bdubbs] - Update to systemd-189. Fixes **#3167**.

- • 2012-09-01

  - ○ [bdubbs] - LFS-7.2 released.

1.4. Resources

1.4.1. FAQ

If during the building of the LFS system you encounter any errors, have any questions, or think there is a typo in the book, please start by consulting the Frequently Asked Questions (FAQ) that is located at **http://www.linuxfromscratch.org/faq/**.

1.4.2. Mailing Lists

The linuxfromscratch.org server hosts a number of mailing lists used for the development of the LFS project. These lists include the main development and support lists, among others. If the FAQ does not solve the problem you are having, the next step would be to search the mailing lists at **http://www.linuxfromscratch.org/search.html**.

For information on the different lists, how to subscribe, archive locations, and additional information, visit **http://www.linuxfromscratch.org/mail.html**.

1.4.3. IRC

Several members of the LFS community offer assistance on Internet Relay Chat (IRC). Before using this support, please make sure that your question is not already answered in the LFS FAQ or the mailing list archives. You can find the IRC network at irc.freenode.net. The support channel is named #LFS-support.

1.4.4. Mirror Sites

The LFS project has a number of world-wide mirrors to make accessing the website and downloading the required packages more convenient. Please visit the LFS website at **http://www.linuxfromscratch.org/mirrors.html** for a list of current mirrors.

1.4.5. Contact Information

Please direct all your questions and comments to one of the LFS mailing lists (see above).

1.5. Help

If an issue or a question is encountered while working through this book, please check the FAQ page at **http://www.linuxfromscratch.org/faq/#generalfaq**. Questions are often already answered there. If your question is not answered on this page, try to find the source of the problem. The following hint will give you some guidance for troubleshooting: **http://www.linuxfromscratch.org/hints /downloads/files/errors.txt**.

If you cannot find your problem listed in the FAQ, search the mailing lists at **http://www.linuxfromscratch.org/search.html**.

We also have a wonderful LFS community that is willing to offer assistance through the mailing lists and IRC (see the Section 1.4, "Resources" section of this book). However, we get several support questions every day and many of them can be easily answered by going to the FAQ and by searching the mailing lists first. So, for us to offer the best assistance possible, you need to do some research on your own first. That allows us to focus on the more unusual support needs. If your searches do not produce a solution, please include all relevant information (mentioned below) in your request for help.

1.5.1. Things to Mention

Apart from a brief explanation of the problem being experienced, the essential things to include in any request for help are:

- The version of the book being used (in this case 7.3)

- The host distribution and version being used to create LFS

- The output from the Section vii, "Host System Requirements"

- The package or section the problem was encountered in

- The exact error message or symptom being received

- Note whether you have deviated from the book at all

> ### Note
>
> Deviating from this book does *not* mean that we will not help you. After all, LFS is about personal preference. Being upfront about any changes to the established procedure helps us evaluate and determine possible causes of your problem.

1.5.2. Configure Script Problems

If something goes wrong while running the **configure** script, review the config.log file. This file may contain errors encountered during **configure** which were not printed to the screen. Include the *relevant* lines if you need to ask for help.

1.5.3. Compilation Problems

Both the screen output and the contents of various files are useful in determining the cause of compilation problems. The screen output from the **configure** script and the **make** run can be helpful. It is not necessary to include the entire output, but do include enough of the relevant information. Below is an example of the type of information to include from the screen output from **make**:

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

In this case, many people would just include the bottom section:

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

This is not enough information to properly diagnose the problem because it only notes that something went wrong, not *what* went wrong. The entire section, as in the example above, is what should be saved because it includes the command that was executed and the associated error message(s).

An excellent article about asking for help on the Internet is available online at **http://catb.org /~esr/faqs/smart-questions.html**. Read and follow the hints in this document to increase the

likelihood of getting the help you need.

Part II. Preparing for the Build

Chapter 2. Preparing a New Partition

2.1. Introduction

In this chapter, the partition which will host the LFS system is prepared. We will create the partition itself, create a file system on it, and mount it.

2.2. Creating a New Partition

Like most other operating systems, LFS is usually installed on a dedicated partition. The recommended approach to building an LFS system is to use an available empty partition or, if you have enough unpartitioned space, to create one.

A minimal system requires a partition of around 2.8 gigabytes (GB). This is enough to store all the source tarballs and compile the packages. However, if the LFS system is intended to be the primary Linux system, additional software will probably be installed which will require additional space. A 10 GB partition is a reasonable size to provide for growth. The LFS system itself will not take up this much room. A large portion of this requirement is to provide sufficient free temporary storage. Compiling packages can require a lot of disk space which will be reclaimed after the package is installed.

Because there is not always enough Random Access Memory (RAM) available for compilation processes, it is a good idea to use a small disk partition as *swap* space. This is used by the kernel to store seldom-used data and leave more memory available for active processes. The *swap* partition for an LFS system can be the same as the one used by the host system, in which case it is not necessary to create another one.

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

> ### Note
>
> For experienced users, other partitioning schemes are possible. The new LFS system can be on a software **RAID** array or an **LVM** logical volume. However, some of these options require an **initramfs**, which is an advanced topic. These partitioning methodologies are not recommended for first time LFS users.

Remember the designation of the new partition (e.g., hda5). This book will refer to this as the LFS partition. Also remember the designation of the *swap* partition. These names will be needed later for the /etc/fstab file.

2.2.1. Other Partition Issues

Requests for advice on system partitioning are often posted on the LFS mailing lists. This is a highly subjective topic. The default for most distributions is to use the entire drive with the exception of one small swap partition. This is not optimal for LFS for several reasons. It reduces flexibility, makes sharing of data across multiple distributions or LFS builds more difficult, makes backups more time consuming, and can waste disk space through inefficient allocation of file system structures.

2.2.1.1. The Root Partition

A root LFS partition (not to be confused with the /root directory) of ten gigabytes is a good compromise for most systems. It provides enough space to build LFS and most of BLFS, but is small enough so that multiple partitions can be easily created for experimentation.

2.2.1.2. The Swap Partition

Most distributions automatically create a swap partition. Generally the recommended size of the swap partition is about twice the amount of physical RAM, however this is rarely needed. If disk space is limited, hold the swap partition to two gigabytes and monitor the amount of disk swapping.

Swapping is never good. Generally you can tell if a system is swapping by just listening to disk activity and observing how the system reacts to commands. The first reaction to swapping should be to check for an unreasonable command such as trying to edit a five gigabyte file. If swapping becomes a normal occurrence, the best solution is to purchase more RAM for your system.

2.2.1.3. Convenience Partitions

There are several other partitions that are not required, but should be considered when designing a disk layout. The following list is not comprehensive, but is meant as a guide.

- /boot – Highly recommended. Use this partition to store kernels and other booting information. To minimize potential boot problems with larger disks, make this the first physical partition on your first disk drive. A partition size of 100 megabytes is quite adequate.

- /home – Highly recommended. Share your home directory and user customization across multiple distributions or LFS builds. The size is generally fairly large and depends on available disk space.

- /usr – A separate /usr partition is generally used if providing a server for a thin client or diskless workstation. It is normally not needed for LFS. A size of five gigabytes will handle most installations.

- /opt – This directory is most useful for BLFS where multiple installations of large packages like Gnome or KDE can be installed without embedding the files in the /usr hierarchy. If used, 5 to 10 gigabytes is generally adequate.

- /tmp – A separate /tmp directory is rare, but useful if configuring a thin client. This partition, if used, will usually not need to exceed a couple of gigabytes.

- /usr/src – This partition is very useful for providing a location to store BLFS source files and share them across LFS builds. It can also be used as a location for building BLFS packages. A reasonably large partition of 30-50 gigabytes allows plenty of room.

Any separate partition that you want automatically mounted upon boot needs to be specified in the /etc/fstab. Details about how to specify partitions will be discussed in Section 8.2, "Creating the /etc/fstab File".

2.3. Creating a File System on the Partition

Now that a blank partition has been set up, the file system can be created. The most widely-used system in the Linux world is the second extended file system (*ext2*), but with newer high-capacity hard disks, journaling file systems are becoming increasingly popular. The third extended filesystem (*ext3*) is a widely used enhancement to *ext2*, which adds journaling capabilities and is compatible with the E2fsprogs utilities. We will create an *ext3* file system. Instructions for creating other file systems can be found at **http://www.linuxfromscratch.org/blfs/view/svn/postlfs/filesystems.html**.

To create an *ext3* file system on the LFS partition, run the following:

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

Replace *<xxx>* with the name of the LFS partition (hda5 in our previous example).

Note

Some host distributions use custom features in their filesystem creation tools (E2fsprogs). This can cause problems when booting into your new LFS in Chapter 9, as those features will not be supported by the LFS-installed E2fsprogs; you will get an error similar to "unsupported filesystem features, upgrade your e2fsprogs". To check if your host system uses custom enhancements, run the following command:

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

If the output contains features other than has_journal, ext_attr, resize_inode, dir_index, filetype, sparse_super, large_file or needs_recovery, then your host system may have custom enhancements. In that case, to avoid later problems, you should compile the stock E2fsprogs package and use the resulting binaries to re-create the filesystem on your LFS partition:

```
cd /tmp
tar -xzvf /path/to/sources/e2fsprogs-1.42.7.tar.gz
cd e2fsprogs-1.42.7
mkdir -v build
cd build
../configure
make #note that we intentionally don't 'make install' here!
./misc/mke2fs -jv /dev/<xxx>
cd /tmp
rm -rfv e2fsprogs-1.42.7
```

If you are using an existing *swap* partition, there is no need to format it. If a new *swap* partition was created, it will need to be initialized with this command:

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

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

2.4. Mounting the New Partition

Now that a file system has been created, the partition needs to be made accessible. In order to do this, the partition needs to be mounted at a chosen mount point. For the purposes of this book, it is assumed that the file system is mounted under /mnt/lfs, but the directory choice is up to you.

Choose a mount point and assign it to the LFS environment variable by running:

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

Next, create the mount point and mount the LFS file system by running:

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

Replace *<xxx>* with the designation of the LFS partition.

If using multiple partitions for LFS (e.g., one for / and another for /usr), mount them using:

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

Replace *<xxx>* and *<yyy>* with the appropriate partition names.

Ensure that this new partition is not mounted with permissions that are too restrictive (such as the nosuid or nodev options). Run the **mount** command without any parameters to see what options are set for the mounted LFS partition. If nosuid, nodev, and/or noatime are set, the partition will need to be remounted.

If you are using a *swap* partition, ensure that it is enabled using the **swapon** command:

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

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

Now that there is an established place to work, it is time to download the packages.

Chapter 3. Packages and Patches

3.1. Introduction

This chapter includes a list of packages that need to be downloaded in order to build a basic Linux system. The listed version numbers correspond to versions of the software that are known to work, and this book is based on their use. We highly recommend against using newer versions because the build commands for one version may not work with a newer version. The newest package versions may also have problems that require work-arounds. These work-arounds will be developed and stabilized in the development version of the book.

Download locations may not always be accessible. If a download location has changed since this book was published, Google (**http://www.google.com/**) provides a useful search engine for most packages. If this search is unsuccessful, try one of the alternative means of downloading discussed at **http://www.linuxfromscratch.org/lfs/packages.html#packages**.

Downloaded packages and patches will need to be stored somewhere that is conveniently available throughout the entire build. A working directory is also required to unpack the sources and build them. $LFS/sources can be used both as the place to store the tarballs and patches and as a working directory. By using this directory, the required elements will be located on the LFS partition and will be available during all stages of the building process.

To create this directory, execute the following command, as user *root*, before starting the download session:

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

Make this directory writable and sticky. "Sticky" means that even if multiple users have write permission on a directory, only the owner of a file can delete the file within a sticky directory. The following command will enable the write and sticky modes:

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

An easy way to download all of the packages and patches is by using **wget-list** as an input to **wget**. For example:

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

Additionally, starting with LFS-7.0, there is a separate file, **md5sums**, can be used to verify that all the correct packages are available before proceeding. Place that file in $LFS/sources and run:

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

3.2. All Packages

Download or otherwise obtain the following packages:

- **Autoconf (2.69) - 1,186 KB:**
  Home page: **http://www.gnu.org/software/autoconf/**
  Download: **http://ftp.gnu.org/gnu/autoconf/autoconf-2.69.tar.xz**
  MD5 sum: 50f97f4159805e374639a73e2636f22e

- **Automake (1.13.1) - 1,392 KB:**
  Home page: **http://www.gnu.org/software/automake/**
  Download: **http://ftp.gnu.org/gnu/automake/automake-1.13.1.tar.xz**
  MD5 sum: a60380ab11e1481376b7747d1b42ced2

- **Bash (4.2) - 6,845 KB:**
  Home page: **http://www.gnu.org/software/bash/**
  Download: **http://ftp.gnu.org/gnu/bash/bash-4.2.tar.gz**
  MD5 sum: 3fb927c7c33022f1c327f14a81c0d4b0

- **Binutils (2.23.1) - 20,953 KB:**
  Home page: **http://www.gnu.org/software/binutils/**

Download: **http://ftp.gnu.org/gnu/binutils/binutils-2.23.1.tar.bz2**
MD5 sum: `33adb18c3048d057ac58d07a3f1adb38`

- **Bison (2.7) - 1,735 KB:**
  Home page: **http://www.gnu.org/software/bison/**
  Download: **http://ftp.gnu.org/gnu/bison/bison-2.7.tar.xz**
  MD5 sum: `234cdfac99257cf99ac4a03c898f37b9`

- **Bzip2 (1.0.6) - 764 KB:**
  Home page: **http://www.bzip.org/**
  Download: **http://www.bzip.org/1.0.6/bzip2-1.0.6.tar.gz**
  MD5 sum: `00b516f4704d4a7cb50a1d97e6e8e15b`

- **Check (0.9.9) - 589 KB:**
  Home page: **http://check.sourceforge.net/**
  Download: **http://sourceforge.net/projects/check/files/check/0.9.9/check-0.9.9.tar.gz**
  MD5 sum: `f3702f2fcfc19ce3f62dca66c241a168`

- **Coreutils (8.21) - 5,248 KB:**
  Home page: **http://www.gnu.org/software/coreutils/**
  Download: **http://ftp.gnu.org/gnu/coreutils/coreutils-8.21.tar.xz**
  MD5 sum: `065ba41828644eca5dd8163446de5d64`

- **DejaGNU (1.5) - 563 KB:**
  Home page: **http://www.gnu.org/software/dejagnu/**
  Download: **http://ftp.gnu.org/gnu/dejagnu/dejagnu-1.5.tar.gz**
  MD5 sum: `3df1cbca885e751e22d3ebd1ac64dc3c`

- **Diffutils (3.2) - 1,976 KB:**
  Home page: **http://www.gnu.org/software/diffutils/**
  Download: **http://ftp.gnu.org/gnu/diffutils/diffutils-3.2.tar.gz**
  MD5 sum: `22e4deef5d8949a727b159d6bc65c1cc`

- **E2fsprogs (1.42.7) - 5,856 KB:**
  Home page: **http://e2fsprogs.sourceforge.net/**
  Download: **http://prdownloads.sourceforge.net/e2fsprogs/e2fsprogs-1.42.7.tar.gz**
  MD5 sum: `a1ec22ef003688dae9f76c74881b22b9`

- **Expect (5.45) - 614 KB:**
  Home page: **http://expect.sourceforge.net/**
  Download: **http://prdownloads.sourceforge.net/expect/expect5.45.tar.gz**
  MD5 sum: `44e1a4f4c877e9ddc5a542dfa7ecc92b`

- **File (5.13) - 627 KB:**
  Home page: **http://www.darwinsys.com/file/**
  Download: **ftp://ftp.astron.com/pub/file/file-5.13.tar.gz**
  MD5 sum: `d60c1364ba956eff7d21f8250808fc6d`

  ### Note

  File (5.13) may no longer be available at the listed location. The site administrators of the master download location occasionally remove older versions when new ones are released. An alternative download location that may have the correct version available can also be found at: **http://www.linuxfromscratch.org/lfs/download.html#ftp**.

- **Findutils (4.4.2) - 2,100 KB:**
  Home page: **http://www.gnu.org/software/findutils/**
  Download: **http://ftp.gnu.org/gnu/findutils/findutils-4.4.2.tar.gz**
  MD5 sum: `351cc4adb07d54877fa15f75fb77d39f`

- **Flex (2.5.37) - 1,280 KB:**
  Home page: **http://flex.sourceforge.net**
  Download: **http://prdownloads.sourceforge.net/flex/flex-2.5.37.tar.bz2**
  MD5 sum: c75940e1fc25108f2a7b3ef42abdae06

- **Gawk (4.0.2) - 1,589 KB:**
  Home page: **http://www.gnu.org/software/gawk/**
  Download: **http://ftp.gnu.org/gnu/gawk/gawk-4.0.2.tar.xz**
  MD5 sum: 8a9b2f1170ac9dcd3eb13716b5ec4021

- **GCC (4.7.2) - 80,942 KB:**
  Home page: **http://gcc.gnu.org/**
  Download: **http://ftp.gnu.org/gnu/gcc/gcc-4.7.2/gcc-4.7.2.tar.bz2**
  MD5 sum: cc308a0891e778cfda7a151ab8a6e762

- **GDBM (1.10) - 640 KB:**
  Home page: **http://www.gnu.org/software/gdbm/**
  Download: **http://ftp.gnu.org/gnu/gdbm/gdbm-1.10.tar.gz**
  MD5 sum: 88770493c2559dc80b561293e39d3570

- **Gettext (0.18.2) - 15,330 KB:**
  Home page: **http://www.gnu.org/software/gettext/**
  Download: **http://ftp.gnu.org/gnu/gettext/gettext-0.18.2.tar.gz**
  MD5 sum: 0c86e5af70c195ab8bd651d17d783928

- **Glibc (2.17) - 10,725 KB:**
  Home page: **http://www.gnu.org/software/libc/**
  Download: **http://ftp.gnu.org/gnu/glibc/glibc-2.17.tar.xz**
  MD5 sum: 87bf675c8ee523ebda4803e8e1cec638

- **GMP (5.1.1) - 1,771 KB:**
  Home page: **http://www.gnu.org/software/gmp/**
  Download: **ftp://ftp.gmplib.org/pub/gmp-5.1.1/gmp-5.1.1.tar.xz**
  MD5 sum: 485b1296e6287fa381e6015b19767989

- **Grep (2.14) - 1,172 KB:**
  Home page: **http://www.gnu.org/software/grep/**
  Download: **http://ftp.gnu.org/gnu/grep/grep-2.14.tar.xz**
  MD5 sum: d4a3f03849d1e17ce56ab76aa5a24cab

- **Groff (1.22.2) - 3,926 KB:**
  Home page: **http://www.gnu.org/software/groff/**
  Download: **http://ftp.gnu.org/gnu/groff/groff-1.22.2.tar.gz**
  MD5 sum: 9f4cd592a5efc7e36481d8d8d8af6d16

- **GRUB (2.00) - 5,016 KB:**
  Home page: **http://www.gnu.org/software/grub/**
  Download: **http://ftp.gnu.org/gnu/grub/grub-2.00.tar.xz**
  MD5 sum: a1043102fbc7bcedbf53e7ee3d17ab91

- **Gzip (1.5) - 704 KB:**
  Home page: **http://www.gnu.org/software/gzip/**
  Download: **http://ftp.gnu.org/gnu/gzip/gzip-1.5.tar.xz**
  MD5 sum: 2a431e169b6f62f7332ef6d47cc53bae

- **Iana-Etc (2.30) - 201 KB:**
  Home page: **http://freshmeat.net/projects/iana-etc/**
  Download: **http://anduin.linuxfromscratch.org/sources/LFS/lfs-packages/conglomeration //iana-etc/iana-etc-2.30.tar.bz2**
  MD5 sum: 3ba3afb1d1b261383d247f46cb135ee8

- **Inetutils (1.9.1) - 1,941 KB:**

Home page: **http://www.gnu.org/software/inetutils/**
Download: **http://ftp.gnu.org/gnu/inetutils/inetutils-1.9.1.tar.gz**
MD5 sum: 944f7196a2b3dba2d400e9088576000c

- **IPRoute2 (3.8.0) - 398 KB:**
  Home page: **http://www.kernel.org/pub/linux/utils/net/iproute2/**
  Download: **http://www.kernel.org/pub/linux/utils/net/iproute2/iproute2-3.8.0.tar.xz**
  MD5 sum: 951622fd770428116dc165acba375414

- **Kbd (1.15.5) - 1,690 KB:**
  Home page: **http://ftp.altlinux.org/pub/people/legion/kbd**
  Download: **http://ftp.altlinux.org/pub/people/legion/kbd/kbd-1.15.5.tar.gz**
  MD5 sum: 34c71feead8ab9c01ec638acea8cd877

- **Kmod (12) - 1,245 KB:**
  Download: **http://www.kernel.org/pub/linux/utils/kernel/kmod/kmod-12.tar.xz**
  MD5 sum: 3d63b146c8ee5a04dfbef4be97f8226b

- **Less (451) - 303 KB:**
  Home page: **http://www.greenwoodsoftware.com/less/**
  Download: **http://www.greenwoodsoftware.com/less/less-451.tar.gz**
  MD5 sum: 765f082658002b2b46b86af4a0da1842

- **LFS-Bootscripts (20130123) - 33 KB:**
  Download: **http://www.linuxfromscratch.org/lfs/downloads/7.3/lfs-bootscripts-20130123.tar.bz2**
  MD5 sum: 2a53fcba68e9f5ed6770c47f05987959

- **Libpipeline (1.2.2) - 733 KB:**
  Home page: **http://libpipeline.nongnu.org/**
  Download: **http://download.savannah.gnu.org/releases/libpipeline/libpipeline-1.2.2.tar.gz**
  MD5 sum: 4367a3f598d171fd43dfa8620ed16d55

- **Libtool (2.4.2) - 2,571 KB:**
  Home page: **http://www.gnu.org/software/libtool/**
  Download: **http://ftp.gnu.org/gnu/libtool/libtool-2.4.2.tar.gz**
  MD5 sum: d2f3b7d4627e69e13514a40e72a24d50

- **Linux (3.8.1) - 69,3292 KB:**
  Home page: **http://www.kernel.org/**
  Download: **http://www.kernel.org/pub/linux/kernel/v3.x/linux-3.8.1.tar.xz**
  MD5 sum: 093b172f907d5455a6746418ad18f5bc

  ### Note

  The Linux kernel is updated relatively often, many times due to discoveries of security vulnerabilities. The latest available 3.8.x kernel version should be used, unless the errata page says otherwise.
  For users with limited speed or expensive bandwidth who wish to update the Linux kernel, a baseline version of the package and patches can be downloaded separately. This may save some time or cost for a subsequent patch level upgrade within a minor release.

- **M4 (1.4.16) - 1,229 KB:**
  Home page: **http://www.gnu.org/software/m4/**
  Download: **http://ftp.gnu.org/gnu/m4/m4-1.4.16.tar.bz2**
  MD5 sum: 8a7cef47fecab6272eb86a6be6363b2f

- **Make (3.82) - 1,213 KB:**
  Home page: **http://www.gnu.org/software/make/**

Download: **http://ftp.gnu.org/gnu/make/make-3.82.tar.bz2**
MD5 sum: `1a11100f3c63fcf5753818e59d63088f`

- **Man-DB (2.6.3) - 1,356 KB:**
  Home page: **http://www.nongnu.org/man-db/**
  Download: **http://download.savannah.gnu.org/releases/man-db/man-db-2.6.3.tar.xz**
  MD5 sum: `a593a095599ae97bcacf8d038659a146`

- **Man-pages (3.47) - 1,108 KB:**
  Home page: **http://www.kernel.org/doc/man-pages/**
  Download: **http://www.kernel.org/pub/linux/docs/man-pages/man-pages-3.47.tar.xz**
  MD5 sum: `2046259794d3003f4dc4dbe4b688ba2f`

- **MPC (1.0.1) - 610 KB:**
  Home page: **http://www.multiprecision.org/**
  Download: **http://www.multiprecision.org/mpc/download/mpc-1.0.1.tar.gz**
  MD5 sum: `b32a2e1a3daa392372fbd586d1ed3679`

- **MPFR (3.1.1) - 1,047 KB:**
  Home page: **http://www.mpfr.org/**
  Download: **http://www.mpfr.org/mpfr-3.1.1/mpfr-3.1.1.tar.xz**
  MD5 sum: `91d51c41fcf2799e4ee7a7126fc95c17`

- **Ncurses (5.9) - 2,760 KB:**
  Home page: **http://www.gnu.org/software/ncurses/**
  Download: **ftp://ftp.gnu.org/gnu/ncurses/ncurses-5.9.tar.gz**
  MD5 sum: `8cb9c412e5f2d96bc6f459aa8c6282a1`

- **Patch (2.7.1) - 660 KB:**
  Home page: **http://savannah.gnu.org/projects/patch/**
  Download: **http://ftp.gnu.org/gnu/patch/patch-2.7.1.tar.xz**
  MD5 sum: `e9ae5393426d3ad783a300a338c09b72`

- **Perl (5.16.2) - 13,403 KB:**
  Home page: **http://www.perl.org/**
  Download: **http://www.cpan.org/src/5.0/perl-5.16.2.tar.bz2**
  MD5 sum: `2818ab01672f005a4e552a713aa27b08`

- **Pkg-config (0.28) - 1,892 KB:**
  Home page: **http://www.freedesktop.org/wiki/Software/pkg-config**
  Download: **http://pkgconfig.freedesktop.org/releases/pkg-config-0.28.tar.gz**
  MD5 sum: `aa3c86e67551adc3ac865160e34a2a0d`

- **Procps (3.3.6) - 528 KB:**
  Home page: **http://sourceforge.net/projects/procps-ng**
  Download: **http://sourceforge.net/projects/procps-ng/files/Production/procps-ng-3.3.6.tar.xz**
  MD5 sum: `0a050d9be531921db3cd38f1371e73e3`

- **Psmisc (22.20) - 422 KB:**
  Home page: **http://psmisc.sourceforge.net/**
  Download: **http://prdownloads.sourceforge.net/psmisc/psmisc-22.20.tar.gz**
  MD5 sum: `a25fc99a6dc7fa7ae6e4549be80b401f`

- **Readline (6.2) - 2,225 KB:**
  Home page: **http://cnswww.cns.cwru.edu/php/chet/readline/rltop.html**
  Download: **http://ftp.gnu.org/gnu/readline/readline-6.2.tar.gz**
  MD5 sum: `67948acb2ca081f23359d0256e9a271c`

- **Sed (4.2.2) - 1,035 KB:**
  Home page: **http://www.gnu.org/software/sed/**
  Download: **http://ftp.gnu.org/gnu/sed/sed-4.2.2.tar.bz2**
  MD5 sum: `7ffe1c7cdc3233e1e0c4b502df253974`

- **Shadow (4.1.5.1) - 2,142 KB:**
  Home page: **http://pkg-shadow.alioth.debian.org/**
  Download: **http://pkg-shadow.alioth.debian.org/releases/shadow-4.1.5.1.tar.bz2**
  MD5 sum: a00449aa439c69287b6d472191dc2247

- **Sysklogd (1.5) - 85 KB:**
  Home page: **http://www.infodrom.org/projects/sysklogd/**
  Download: **http://www.infodrom.org/projects/sysklogd/download/sysklogd-1.5.tar.gz**
  MD5 sum: e053094e8103165f98ddafe828f6ae4b

- **Sysvinit (2.88dsf) - 108 KB:**
  Home page: **http://savannah.nongnu.org/projects/sysvinit**
  Download: **http://download.savannah.gnu.org/releases/sysvinit/sysvinit-2.88dsf.tar.bz2**
  MD5 sum: 6eda8a97b86e0a6f59dabbf25202aa6f

- **Tar (1.26) - 2,285 KB:**
  Home page: **http://www.gnu.org/software/tar/**
  Download: **http://ftp.gnu.org/gnu/tar/tar-1.26.tar.bz2**
  MD5 sum: 2cee42a2ff4f1cd4f9298eeeb2264519

- **Tcl (8.6.0) - 8,435 KB:**
  Home page: **http://tcl.sourceforge.net/**
  Download: **http://prdownloads.sourceforge.net/tcl/tcl8.6.0-src.tar.gz**
  MD5 sum: 573aa5fe678e9185ef2b3c56b24658d3

- **Time Zone Data (2012j) - 209 KB:**
  Home page: **http://www.iana.org/time-zones**
  Download: **http://www.iana.org//time-zones/repository/releases/tzdata2012j.tar.gz**
  MD5 sum: ba2f92ae7ad099090e8f86cff2f2d799

- **Texinfo (5.0) - 3,000 KB:**
  Home page: **http://www.gnu.org/software/texinfo/**
  Download: **http://ftp.gnu.org/gnu/texinfo/texinfo-5.0.tar.xz**
  MD5 sum: ef2fad34c71ddc95b20c7d6a08c0d7a6

- **Systemd (197) - 2,012 KB:**
  Home page: **http://www.freedesktop.org/wiki/Software/systemd/**
  Download: **http://www.freedesktop.org/software/systemd/systemd-197.tar.xz**
  MD5 sum: 56a860dceadfafe59f40141eb5223743

- **Udev-lfs Tarball (197) - 17 KB:**
  Download: **http://anduin.linuxfromscratch.org/sources/other/udev-lfs-197-2.tar.bz2**
  MD5 sum: f4272c121514caf0c2a6245fbffeb047

- **Util-linux (2.22.2) - 3,028 KB:**
  Home page: **http://userweb.kernel.org/~kzak/util-linux/**
  Download: **http://www.kernel.org/pub/linux/utils/util-linux/v2.22/util-linux-2.22.2.tar.xz**
  MD5 sum: eeacbfdd2556acd899a2d0ffdb446185

- **Vim (7.3) - 8,675 KB:**
  Home page: **http://www.vim.org**
  Download: **ftp://ftp.vim.org/pub/vim/unix/vim-7.3.tar.bz2**
  MD5 sum: 5b9510a17074e2b37d8bb38ae09edbf2

- **Xz Utils (5.0.4) - 894 KB:**
  Home page: **http://tukaani.org/xz**
  Download: **http://tukaani.org/xz/xz-5.0.4.tar.xz**
  MD5 sum: 161015c4a65b1f293d31810e1df93090

- **Zlib (1.2.7) - 493 KB:**
  Home page: **http://www.zlib.net/**
  Download: **http://www.zlib.net/zlib-1.2.7.tar.bz2**

MD5 sum: `2ab442d169156f34c379c968f3f482dd`

Total size of these packages: about 915 MB

3.3. Needed Patches

In addition to the packages, several patches are also required. These patches correct any mistakes in the packages that should be fixed by the maintainer. The patches also make small modifications to make the packages easier to work with. The following patches will be needed to build an LFS system:

- **Bash Upstream Fixes Patch - 55 KB:**
  Download: **http://www.linuxfromscratch.org/patches/lfs/7.3/bash-4.2-fixes-11.patch**
  MD5 sum: `366672f68e7cf214bdbef2ef14b13c23`

- **Binutils Test Suite Patch - 2.0 KB:**
  Download: **http://www.linuxfromscratch.org/patches/lfs/7.3/binutils-2.23.1-testsuite_fix-1.patch**
  MD5 sum: `cb47fae1bc572d45f4b0cff8ae8ecba8`

- **Bzip2 Documentation Patch - 1.6 KB:**
  Download: **http://www.linuxfromscratch.org/patches/lfs/7.3/bzip2-1.0.6-install_docs-1.patch**
  MD5 sum: `6a5ac7e89b791aae556de0f745916f7f`

- **Coreutils Internationalization Fixes Patch - 132 KB:**
  Download: **http://www.linuxfromscratch.org/patches/lfs/7.3/coreutils-8.21-i18n-1.patch**
  MD5 sum: `ada0ea6e1c00c4b7e0d634f49827943e`

- **Flex Regression Tests Patch - 2.8 KB:**
  Download: **http://www.linuxfromscratch.org/patches/lfs/7.3/flex-2.5.37-bison-2.6.1-1.patch**
  MD5 sum: `d5b001ef9bdbbe32e2f27576d97d8ff0`

- **Kbd Backspace/Delete Fix Patch - 12 KB:**
  Download: **http://www.linuxfromscratch.org/patches/lfs/7.3/kbd-1.15.5-backspace-1.patch**
  MD5 sum: `f75cca16a38da6caa7d52151f7136895`

- **Make Upstream Fixes Patch - 10 KB:**
  Download: **http://www.linuxfromscratch.org/patches/lfs/7.3/make-3.82-upstream_fixes-3.patch**
  MD5 sum: `95027ab5b53d01699845d9b7e1dc878d`

- **Perl Libc Patch - 1.6 KB:**
  Download: **http://www.linuxfromscratch.org/patches/lfs/7.3/perl-5.16.2-libc-1.patch**
  MD5 sum: `daf5c64fd7311e924966842680535f8f`

- **Readline Upstream Fixes Patch - 1.3 KB:**
  Download: **http://www.linuxfromscratch.org/patches/lfs/7.3/readline-6.2-fixes-1.patch**
  MD5 sum: `3c185f7b76001d3d0af614f6f2cd5dfa`

Total size of these patches: about 218.3 KB

In addition to the above required patches, there exist a number of optional patches created by the LFS community. These optional patches solve minor problems or enable functionality that is not enabled by default. Feel free to peruse the patches database located at **http://www.linuxfromscratch.org/patches/downloads/** and acquire any additional patches to suit your system needs.

Chapter 4. Final Preparations

4.1. About $LFS

Throughout this book, the environment variable LFS will be used. It is paramount that this variable is always defined. It should be set to the mount point chosen for the LFS partition. Check that the LFS variable is set up properly with:

```
echo $LFS
```

Make sure the output shows the path to the LFS partition's mount point, which is /mnt/lfs if the provided example was followed. If the output is incorrect, the variable can be set with:

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

Having this variable set is beneficial in that commands such as **mkdir $LFS/tools** can be typed literally. The shell will automatically replace "$LFS" with "/mnt/lfs" (or whatever the variable was set to) when it processes the command line.

Do not forget to check that $LFS is set whenever you leave and reenter the current working environment (as when doing a **su** to *root* or another user).

4.2. Creating the $LFS/tools Directory

All programs compiled in Chapter 5 will be installed under $LFS/tools to keep them separate from the programs compiled in Chapter 6. The programs compiled here are temporary tools and will not be a part of the final LFS system. By keeping these programs in a separate directory, they can easily be discarded later after their use. This also prevents these programs from ending up in the host production directories (easy to do by accident in Chapter 5).

Create the required directory by running the following as *root*:

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

The next step is to create a /tools symlink on the host system. This will point to the newly-created directory on the LFS partition. Run this command as *root* as well:

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

Note

The above command is correct. The **ln** command has a few syntactic variations, so be sure to check **info coreutils ln** and ln(1) before reporting what you may think is an error.

The created symlink enables the toolchain to be compiled so that it always refers to /tools, meaning that the compiler, assembler, and linker will work both in Chapter 5 (when we are still using some tools from the host) and in the next (when we are "chrooted" to the LFS partition).

4.3. Adding the LFS User

When logged in as user *root*, making a single mistake can damage or destroy a system. Therefore, we recommend building the packages in this chapter as an unprivileged user. You could use your own user name, but to make it easier to set up a clean working environment, create a new user called *lfs* as a member of a new group (also named *lfs*) and use this user during the installation process. As *root*, issue the following commands to add the new user:

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

The meaning of the command line options:

-s /bin/bash
> This makes **bash** the default shell for user *lfs*.

-g lfs
> This option adds user *lfs* to group *lfs*.

-m
> This creates a home directory for *lfs*.

-k /dev/null
> This parameter prevents possible copying of files from a skeleton directory (default is /etc/skel) by changing the input location to the special null device.

lfs
> This is the actual name for the created group and user.

To log in as *lfs* (as opposed to switching to user *lfs* when logged in as *root*, which does not require the *lfs* user to have a password), give *lfs* a password:

```
passwd lfs
```

Grant *lfs* full access to $LFS/tools by making *lfs* the directory owner:

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

If a separate working directory was created as suggested, give user *lfs* ownership of this directory:

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

Next, login as user *lfs*. This can be done via a virtual console, through a display manager, or with the following substitute user command:

```
su - lfs
```

The "-" instructs **su** to start a login shell as opposed to a non-login shell. The difference between these two types of shells can be found in detail in bash(1) and **info bash**.

4.4. Setting Up the Environment

Set up a good working environment by creating two new startup files for the **bash** shell. While logged in as user *lfs*, issue the following command to create a new .bash_profile:

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

When logged on as user *lfs*, the initial shell is usually a *login* shell which reads the /etc/profile of the host (probably containing some settings and environment variables) and then .bash_profile. The **exec env -i.../bin/bash** command in the .bash_profile file replaces the running shell with a new one with a completely empty environment, except for the HOME, TERM, and PS1 variables. This ensures that no unwanted and potentially hazardous environment variables from the host system leak into the build environment. The technique used here achieves the goal of ensuring a clean environment.

The new instance of the shell is a *non-login* shell, which does not read the /etc/profile or .bash_profile files, but rather reads the .bashrc file instead. Create the .bashrc file now:

```
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
```

The **set +h** command turns off **bash**'s hash function. Hashing is ordinarily a useful feature—**bash** uses a hash table to remember the full path of executable files to avoid searching the PATH time and again to find the same executable. However, the new tools should be used as soon as they are installed. By switching off the hash function, the shell will always search the PATH when a program is to be run. As such, the shell will find the newly compiled tools in $LFS/tools as soon as they are available without remembering a previous version of the same program in a different location.

Setting the user file-creation mask (umask) to 022 ensures that newly created files and directories are only writable by their owner, but are readable and executable by anyone (assuming default modes are used by the open(2) system call, new files will end up with permission mode 644 and directories with mode 755).

The LFS variable should be set to the chosen mount point.

The LC_ALL variable controls the localization of certain programs, making their messages follow the conventions of a specified country. If the host system uses a version of Glibc older than 2.2.4, having LC_ALL set to something other than "POSIX" or "C" (during this chapter) may cause issues if you exit the chroot environment and wish to return later. Setting LC_ALL to "POSIX" or "C" (the two are equivalent) ensures that everything will work as expected in the chroot environment.

The LFS_TGT variable sets a non-default, but compatible machine description for use when building our cross compiler and linker and when cross compiling our temporary toolchain. More information is contained in Section 5.2, "Toolchain Technical Notes".

By putting /tools/bin ahead of the standard PATH, all the programs installed in Chapter 5 are picked up by the shell immediately after their installation. This, combined with turning off hashing, limits the risk that old programs are used from the host when the same programs are available in the chapter 5 environment.

Finally, to have the environment fully prepared for building the temporary tools, source the just-created user profile:

```
source ~/.bash_profile
```

4.5. About SBUs

Many people would like to know beforehand approximately how long it takes to compile and install each package. Because Linux From Scratch can be built on many different systems, it is impossible to provide accurate time estimates. The biggest package (Glibc) will take approximately 20 minutes on the fastest systems, but could take up to three days on slower systems! Instead of providing actual times, the Standard Build Unit (SBU) measure will be used instead.

The SBU measure works as follows. The first package to be compiled from this book is Binutils in Chapter 5. The time it takes to compile this package is what will be referred to as the Standard Build Unit or SBU. All other compile times will be expressed relative to this time.

For example, consider a package whose compilation time is 4.5 SBUs. This means that if a system took 10 minutes to compile and install the first pass of Binutils, it will take *approximately* 45 minutes to build this example package. Fortunately, most build times are shorter than the one for Binutils.

In general, SBUs are not entirely accurate because they depend on many factors, including the host system's version of GCC. They are provided here to give an estimate of how long it might take to install a package, but the numbers can vary by as much as dozens of minutes in some cases.

To view actual timings for a number of specific machines, we recommend The LinuxFromScratch SBU Home Page at **http://www.linuxfromscratch.org/~sbu/**.

Note

For many modern systems with multiple processors (or cores) the compilation time for a package can be reduced by performing a "parallel make" by either setting an environment variable or telling the **make** program how many processors are available. For instance, a Core2Duo can support two simultaneous processes with:

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

or just building with:

```
make -j2
```

When multiple processors are used in this way, the SBU units in the book will vary even more than they normally would. Analyzing the output of the build process will also be more difficult because the lines of different processes will be interleaved. If you run into a problem with a build step, revert back to a single processor build to properly analyze the error messages.

4.6. About the Test Suites

Most packages provide a test suite. Running the test suite for a newly built package is a good idea

because it can provide a "sanity check" indicating that everything compiled correctly. A test suite that passes its set of checks usually proves that the package is functioning as the developer intended. It does not, however, guarantee that the package is totally bug free.

Some test suites are more important than others. For example, the test suites for the core toolchain packages—GCC, Binutils, and Glibc—are of the utmost importance due to their central role in a properly functioning system. The test suites for GCC and Glibc can take a very long time to complete, especially on slower hardware, but are strongly recommended.

> ### Note
>
> Experience has shown that there is little to be gained from running the test suites in Chapter 5. There can be no escaping the fact that the host system always exerts some influence on the tests in that chapter, often causing inexplicable failures. Because the tools built in Chapter 5 are temporary and eventually discarded, we do not recommend running the test suites in Chapter 5 for the average reader. The instructions for running those test suites are provided for the benefit of testers and developers, but they are strictly optional.

A common issue with running the test suites for Binutils and GCC is running out of pseudo terminals (PTYs). This can result in a high number of failing tests. This may happen for several reasons, but the most likely cause is that the host system does not have the *devpts* file system set up correctly. This issue is discussed in greater detail at **http://www.linuxfromscratch.org//lfs/faq.html#no-ptys**.

Sometimes package test suites will fail, but for reasons which the developers are aware of and have deemed non-critical. Consult the logs located at **http://www.linuxfromscratch.org/lfs/build-logs/7.3/** to verify whether or not these failures are expected. This site is valid for all tests throughout this book.

Chapter 5. Constructing a Temporary System

5.1. Introduction

This chapter shows how to build a minimal Linux system. This system will contain just enough tools to start constructing the final LFS system in Chapter 6 and allow a working environment with more user convenience than a minimum environment would.

There are two steps in building this minimal system. The first step is to build a new and host-independent toolchain (compiler, assembler, linker, libraries, and a few useful utilities). The second step uses this toolchain to build the other essential tools.

The files compiled in this chapter will be installed under the $LFS/tools directory to keep them separate from the files installed in the next chapter and the host production directories. Since the packages compiled here are temporary, we do not want them to pollute the soon-to-be LFS system.

5.2. Toolchain Technical Notes

This section explains some of the rationale and technical details behind the overall build method. It is not essential to immediately understand everything in this section. Most of this information will be

clearer after performing an actual build. This section can be referred to at any time during the process.

The overall goal of Chapter 5 is to produce a temporary area that contains a known-good set of tools that can be isolated from the host system. By using **chroot**, the commands in the remaining chapters will be contained within that environment, ensuring a clean, trouble-free build of the target LFS system. The build process has been designed to minimize the risks for new readers and to provide the most educational value at the same time.

> ### Note
>
> Before continuing, be aware of the name of the working platform, often referred to as the target triplet. A simple way to determine the name of the target triplet is to run the **config.guess** script that comes with the source for many packages. Unpack the Binutils sources and run the script: **./config.guess** and note the output. For example, for a modern 32-bit Intel processor the output will likely be *i686-pc-linux-gnu*.
>
> Also be aware of the name of the platform's dynamic linker, often referred to as the dynamic loader (not to be confused with the standard linker **ld** that is part of Binutils). The dynamic linker provided by Glibc finds and loads the shared libraries needed by a program, prepares the program to run, and then runs it. The name of the dynamic linker for a 32-bit Intel machine will be ld-linux.so.2. A sure-fire way to determine the name of the dynamic linker is to inspect a random binary from the host system by running: **readelf -l <name of binary> | grep interpreter** and noting the output. The authoritative reference covering all platforms is in the shlib-versions file in the root of the Glibc source tree.

Some key technical points of how the Chapter 5 build method works:

* Slightly adjusting the name of the working platform, by changing the "vendor" field target triplet by way of the LFS_TGT variable, ensures that the first build of Binutils and GCC produces a compatible cross-linker and cross-compiler. Instead of producing binaries for another architecture, the cross-linker and cross-compiler will produce binaries compatible with the current hardware.

* The temporary libraries are cross-compiled. Because a cross-compiler by its nature cannot rely on anything from its host system, this method removes potential contamination of the target system by lessening the chance of headers or libraries from the host being incorporated into the new tools. Cross-compilation also allows for the possibility of building both 32-bit and 64-bit libraries on 64-bit capable hardware.

* Careful manipulation of the GCC source tells the compiler which target dynamic linker will be used.

Binutils is installed first because the **configure** runs of both GCC and Glibc perform various feature tests on the assembler and linker to determine which software features to enable or disable. This is more important than one might first realize. An incorrectly configured GCC or Glibc can result in a subtly broken toolchain, where the impact of such breakage might not show up until near the end of the build of an entire distribution. A test suite failure will usually highlight this error before too much additional work is performed.

Binutils installs its assembler and linker in two locations, /tools/bin and /tools/$LFS_TGT/bin. The tools in one location are hard linked to the other. An important facet of the linker is its library search order. Detailed information can be obtained from **ld** by passing it the *--verbose* flag. For example, an **ld --verbose | grep SEARCH** will illustrate the current search paths and their order. It

shows which files are linked by `ld` by compiling a dummy program and passing the `--verbose` switch to the linker. For example, `gcc dummy.c -Wl,--verbose 2>&1 | grep succeeded` will show all the files successfully opened during the linking.

The next package installed is GCC. An example of what can be seen during its run of `configure` is:

```
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.

Next installed are sanitized Linux API headers. These allow the standard C library (Glibc) to interface with features that the Linux kernel will provide.

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.

During the second pass of Binutils, we are able to utilize the `--with-lib-path` configure switch to control `ld`'s library search path.

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. From this point onwards, the core toolchain is self-contained and self-hosted. The remainder of the [Chapter 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. General Compilation Instructions

When building packages there are several assumptions made within the instructions:

- Several of the packages are patched before compilation, but only when the patch is needed to circumvent a problem. A patch is often needed in both this and the next chapter, but sometimes in only one or the other. Therefore, do not be concerned if instructions for a downloaded patch seem to be missing. Warning messages about *offset* or *fuzz* may also be encountered when

applying a patch. Do not worry about these warnings, as the patch was still successfully applied.

- During the compilation of most packages, there will be several warnings that scroll by on the screen. These are normal and can safely be ignored. These warnings are as they appear —warnings about deprecated, but not invalid, use of the C or C++ syntax. C standards change fairly often, and some packages still use the older standard. This is not a problem, but does prompt the warning.

- Check one last time that the LFS environment variable is set up properly:

```
echo $LFS
```

Make sure the output shows the path to the LFS partition's mount point, which is /mnt/lfs, using our example.

- Finally, two last important items must be emphasized:

Important

The build instructions assume that the Host System Requirements, including symbolic links, have been set properly:

- **bash** is the shell in use.

- **sh** is a symbolic link to **bash**.

- **/usr/bin/awk** is a symbolic link to **gawk**.

- **/usr/bin/yacc** is a symbolic link to **bison** or a small script that executes bison.

Important

To re-emphasize the build process:

1. Place all the sources and patches in a directory that will be accessible from the chroot environment such as /mnt/lfs/sources/. Do *not* put sources in /mnt/lfs/tools/.

2. Change to the sources directory.

3. For each package:

   a. Using the **tar** program, extract the package to be built. In Chapter 5, ensure you are the *lfs* user when extracting the package.

   b. Change to the directory created when the package was extracted.

   c. Follow the book's instructions for building the package.

   d. Change back to the sources directory.

   e. Delete the extracted source directory and any *<package>*-build directories that were created in the build process unless instructed otherwise.

5.4. Binutils-2.23.1 - Pass 1

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

Approximate build time:	1 SBU
Required disk space:	391 MB

5.4.1. Installation of Cross Binutils

Note

Go back and re-read the notes in the previous section. Understanding the notes labeled important will save you a lot of problems later.

It is important that Binutils be the first package compiled because both Glibc and GCC perform various tests on the available linker and assembler to determine which of their own features to enable.

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

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

Note

In order for the SBU values listed in the rest of the book to be of any use, measure the time it takes to build this package from the configuration, up to and including the first install. To achieve this easily, wrap the commands in a **time** command like this: **time { ./configure ... && ... && make install; }**.

Note

The approximate build SBU values and required disk space in Chapter 5 does not include test suite data.

Now prepare Binutils for compilation:

```
../binutils-2.23.1/configure     \
    --prefix=/tools            \
    --with-sysroot=$LFS        \
    --with-lib-path=/tools/lib \
    --target=$LFS_TGT          \
    --disable-nls              \
    --disable-werror
```

The meaning of the configure options:

--prefix=/tools

This tells the configure script to prepare to install the Binutils programs in the /tools directory.

--with-sysroot=$LFS

> For cross compilation, this tells the build system to look in $LFS for the target system libraries as needed.

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

> This specifies which library path the linker should be configured to use.

--target=$LFS_TGT

> Because the machine description in the LFS_TGT variable is slightly different than the value returned by the **config.guess** script, this switch will tell the **configure** script to adjust Binutil's build system for building a cross linker.

--disable-nls

> This disables internationalization as i18n is not needed for the temporary tools.

--disable-werror

> This prevents the build from stopping in the event that there are warnings from the host's compiler.

Continue with compiling the package:

```
make
```

Compilation is now complete. Ordinarily we would now run the test suite, but at this early stage the test suite framework (Tcl, Expect, and DejaGNU) is not yet in place. The benefits of running the tests at this point are minimal since the programs from this first pass will soon be replaced by those from the second.

If building on x86_64, create a symlink to ensure the sanity of the toolchain:

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

Install the package:

```
make install
```

Details on this package are located in <u>Section 6.13.2, "Contents of Binutils."</u>

5.5. GCC-4.7.2 - Pass 1

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

Approximate build time:	5.5 SBU
Required disk space:	1.4 GB

5.5.1. Installation of Cross GCC

GCC now requires the GMP, MPFR and MPC packages. As these packages may not be included in your host distribution, they will be built with GCC. Unpack each package into the GCC source directory and rename the resulting directories so the GCC build procedures will automatically use them:

> **Note**
>
> There are frequent misunderstandings about this chapter. The procedures are the same as
> every other chapter as explained earlier (Package build instructions). First extract the gcc
> tarball from the sources directory and then change to the directory created. Only then
> should you proceed with the instructions below.

```
tar -Jxf ../mpfr-3.1.1.tar.xz
mv -v mpfr-3.1.1 mpfr
tar -Jxf ../gmp-5.1.1.tar.xz
mv -v gmp-5.1.1 gmp
tar -zxf ../mpc-1.0.1.tar.gz
mv -v mpc-1.0.1 mpc
```

The following command will change the location of GCC's default dynamic linker to use the one
installed in /tools. It also removes /usr/include from GCC's include search path. Issue:

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

In case the above seems hard to follow, let's break it down a bit. First we find all the files under the
gcc/config directory that are named either linux.h, linux64.h or sysv4.h. For each file found,
we copy it to a file of the same name but with an added suffix of ".orig". Then the first sed
expression prepends "/tools" to every instance of "/lib/ld", "/lib64/ld" or "/lib32/ld", while the
second one replaces hard-coded instances of "/usr". Next, we add our define statements which alter
the default startfile prefix to the end of the file. Note that the trailing "/" in "/tools/lib/" is required.
Finally, we use **touch** to update the timestamp on the copied files. When used in conjunction with **cp
-u**, this prevents unexpected changes to the original files in case the commands are inadvertently run
twice.

GCC doesn't detect stack protection correctly, which causes problems for the build of Glibc-2.17, so
fix that by issuing the following command:

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

Do not build the .info files. They are not needed here and are broken with the current version of
makeinfo.

```
sed -i 's/BUILD_INFO=info/BUILD_INFO=/' gcc/configure
```

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-4.7.2/configure            \
    --target=$LFS_TGT             \
    --prefix=/tools               \
    --with-sysroot=$LFS           \
    --with-newlib                 \
    --without-headers             \
    --with-local-prefix=/tools \
    --with-native-system-header-dir=/tools/include \
    --disable-nls                 \
    --disable-shared              \
    --disable-multilib            \
    --disable-decimal-float       \
    --disable-threads             \
    --disable-libmudflap          \
    --disable-libssp              \
    --disable-libgomp             \
    --disable-libquadmath         \
    --enable-languages=c          \
    --with-mpfr-include=$(pwd)/../gcc-4.7.2/mpfr/src \
    --with-mpfr-lib=$(pwd)/mpfr/src/.libs
```

The meaning of the configure options:

--with-newlib
> Since a working C library is not yet available, this ensures that the inhibit_libc constant is defined when building libgcc. This prevents the compiling of any code that requires libc support.

--without-headers
> When creating a complete cross-compiler, GCC requires standard headers compatible with the target system. For our purposes these headers will not be needed. This switch prevents GCC from looking for them.

--with-local-prefix=/tools
> The local prefix is the location in the system that GCC will search for locally installed include files. The default is /usr/local. Setting this to /tools helps keep the host location of /usr/local out of this GCC's search path.

--with-native-system-header-dir=/tools/include
> By default GCC searches /usr/include for system headers. In conjunction with the sysroot switch, this would translate normally to $LFS/usr/include. However the headers that will be installed in the next two sections will go to $LFS/tools/include. This switch ensures that gcc will find them correctly. In the second pass of GCC, this same switch will ensure that no headers from the host system are found.

--disable-shared
> This switch forces GCC to link its internal libraries statically. We do this to avoid possible issues with the host system.

--disable-decimal-float, --disable-threads, --disable-libmudflap, --disable-libssp, --disable-libgomp, --disable-libquadmath
> These switches disable support for the decimal floating point extension, threading, libmudflap, libssp and libgomp and libquadmath respectively. These features will fail to compile when building a cross-compiler and are not necessary for the task of cross-compiling the temporary

libc.

--disable-multilib
>   On x86_64, LFS does not yet support a multilib configuration. This switch is harmless for x86.

--enable-languages=c
>   This option ensures that only the C compiler is built. This is the only language needed now.

Compile GCC by running:

```
make
```

Compilation is now complete. At this point, the test suite would normally be run, but, as mentioned before, the test suite framework is not in place yet. The benefits of running the tests at this point are minimal since the programs from this first pass will soon be replaced.

Install the package:

```
make install
```

Using *--disable-shared* means that the libgcc_eh.a file isn't created and installed. The Glibc package depends on this library as it uses *-lgcc_eh* within its build system. This dependency can be satisfied by creating a symlink to libgcc.a, since that file will end up containing the objects normally contained in libgcc_eh.a:

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

Details on this package are located in [Section 6.17.2, "Contents of GCC."](#)

5.6. Linux-3.8.1 API Headers

The Linux API Headers (in linux-3.8.1.tar.xz) expose the kernel's API for use by Glibc.

Approximate build time:	0.1 SBU
Required disk space:	511 MB

5.6.1. 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.

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

Details on this package are located in [Section 6.7.2, "Contents of Linux API Headers."](#)

5.7. Glibc-2.17

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.

Approximate build time:	5.4 SBU
Required disk space:	554 MB

5.7.1. Installation of Glibc

In some cases, particularly LFS 7.1, the rpc headers were not installed properly. Test to see if they are installed in the host system and install if they are not:

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

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

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

Next, prepare Glibc for compilation:

```
../glibc-2.17/configure                              \
      --prefix=/tools                                \
      --host=$LFS_TGT                                \
      --build=$(../glibc-2.17/scripts/config.guess) \
      --disable-profile                              \
```

```
        --enable-kernel=2.6.25                            \
        --with-headers=/tools/include                     \
        libc_cv_forced_unwind=yes                         \
        libc_cv_ctors_header=yes                          \
        libc_cv_c_cleanup=yes
```

The meaning of the configure options:

--host=$LFS_TGT, --build=$(../glibc-2.17/scripts/config.guess)
> The combined effect of these switches is that Glibc's build system configures itself to cross-compile, using the cross-linker and cross-compiler in /tools.

--disable-profile
> This builds the libraries without profiling information. Omit this option if profiling on the temporary tools is necessary.

--enable-kernel=2.6.25
> This tells Glibc to compile the library with support for 2.6.25 and later Linux kernels. Workarounds for older kernels are not enabled.

--with-headers=/tools/include
> This tells Glibc to compile itself against the headers recently installed to the tools directory, so that it knows exactly what features the kernel has and can optimize itself accordingly.

libc_cv_forced_unwind=yes
> The linker installed during Section 5.4, "Binutils-2.23.1 - Pass 1" was cross-compiled and as such cannot be used until Glibc has been installed. This means that the configure test for force-unwind support will fail, as it relies on a working linker. The libc_cv_forced_unwind=yes variable is passed in order to inform **configure** that force-unwind support is available without it having to run the test.

libc_cv_c_cleanup=yes
> Simlarly, we pass libc_cv_c_cleanup=yes through to the **configure** script so that the test is skipped and C cleanup handling support is configured.

libc_cv_ctors_header=yes
> Simlarly, we pass libc_cv_ctors_header=yes through to the **configure** script so that the test is skipped and gcc constructor support is configured.

During this stage the following warning might appear:

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

The missing or incompatible **msgfmt** program is generally harmless. This **msgfmt** program is part of the Gettext package which the host distribution should provide.

Compile the package:

```
make
```

This package does come with a test suite, however, it cannot be run at this time because we do not have a C++ compiler yet.

Note

The test suite also requires locale data to be installed in order to run successfully. Locale data provides information to the system regarding such things as the date, time, and currency formats accepted and output by system utilities. If the test suites are not being run in this chapter (as per the recommendation), there is no need to install the locales now. The appropriate locales will be installed in the next chapter. To install the Glibc locales anyway, use instructions from Section 6.9, "Glibc-2.17."

Install the package:

```
make install
```

Caution

At this point, it is imperative to stop and ensure that the basic functions (compiling and linking) of the new toolchain are working as expected. To perform a sanity check, run the following commands:

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

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

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

Note that /tools/lib, or /tools/lib64 for 64-bit machines appears as the prefix of the dynamic linker.

If the output is not shown as above or there was no output at all, then something is wrong. Investigate and retrace the steps to find out where the problem is and correct it. This issue must be resolved before continuing on.

Once all is well, clean up the test files:

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

Note

Building Binutils in the next section will serve as an additional check that the toolchain has been built properly. If Binutils fails to build, it is an indication that something has gone wrong with the previous Binutils, GCC, or Glibc installations.

Details on this package are located in Section 6.9.4, "Contents of Glibc."

5.8. Binutils-2.23.1 - Pass 2

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

Approximate build time:	1.1 SBU
Required disk space:	407 MB

5.8.1. Installation of Binutils

Create a separate build directory again:

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

Prepare Binutils for compilation:

```
CC=$LFS_TGT-gcc            \
AR=$LFS_TGT-ar            \
RANLIB=$LFS_TGT-ranlib     \
../binutils-2.23.1/configure \
    --prefix=/tools        \
    --disable-nls          \
    --with-lib-path=/tools/lib
```

The meaning of the new configure options:

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

  Because this is really a native build of Binutils, setting these variables ensures that the build system uses the cross-compiler and associated tools instead of the ones on the host system.

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

  This tells the configure script to specify the library search path during the compilation of Binutils, resulting in /tools/lib being passed to the linker. This prevents the linker from searching through library directories on the host.

Compile the package:

```
make
```

Install the package:

```
make install
```

Now prepare the linker for the "Re-adjusting" phase in the next chapter:

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

The meaning of the make parameters:

-C ld clean

  This tells the make program to remove all compiled files in the ld subdirectory.

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

This option rebuilds everything in the ld subdirectory. Specifying the LIB_PATH Makefile variable on the command line allows us to override the default value of the temporary tools and point it to the proper final path. The value of this variable specifies the linker's default library search path. This preparation is used in the next chapter.

Details on this package are located in <u>Section 6.13.2, "Contents of Binutils."</u>

5.9. GCC-4.7.2 - Pass 2

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

Approximate build time:	7.1 SBU
Required disk space:	1.8 GB

5.9.1. Installation of GCC

Our first build of GCC has installed a couple of internal system headers. Normally one of them, limits.h will in turn include the corresponding system limits.h header, in this case, /tools /include/limits.h. However, at the time of the first build of gcc /tools/include/limits.h did not exist, so the internal header that GCC installed is a partial, self-contained file and does not include the extended features of the system header. This was adequate for building the temporary libc, but this build of GCC now requires the full internal header. Create a full version of the internal header using a command that is identical to what the GCC build system does in normal circumstances:

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

For x86 machines, a bootstrap build of GCC uses the -fomit-frame-pointer compiler flag. Non-bootstrap builds omit this flag by default, and the goal should be to produce a compiler that is exactly the same as if it were bootstrapped. Apply the following **sed** command to force the build to use the flag:

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

Once again, change the location of GCC's default dynamic linker to use the one installed in /tools.

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

As in the first build of GCC it requires the GMP, MPFR and MPC packages. Unpack the tarballs and move them into the required directory names:

```
tar -Jxf ../mpfr-3.1.1.tar.xz
mv -v mpfr-3.1.1 mpfr
tar -Jxf ../gmp-5.1.1.tar.xz
mv -v gmp-5.1.1 gmp
tar -zxf ../mpc-1.0.1.tar.gz
mv -v mpc-1.0.1 mpc
```

Again, do not build the .info files. They are not needed here and are broken with the current version of **makeinfo**.

```
sed -i 's/BUILD_INFO=info/BUILD_INFO=/' gcc/configure
```

Create a separate build directory again:

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

Before starting to build GCC, remember to unset any environment variables that override the default optimization flags.

Now prepare GCC for compilation:

```
CC=$LFS_TGT-gcc \
AR=$LFS_TGT-ar                  \
RANLIB=$LFS_TGT-ranlib          \
../gcc-4.7.2/configure          \
    --prefix=/tools             \
    --with-local-prefix=/tools  \
    --with-native-system-header-dir=/tools/include \
    --enable-clocale=gnu        \
    --enable-shared             \
    --enable-threads=posix      \
    --enable-__cxa_atexit       \
    --enable-languages=c,c++    \
    --disable-libstdcxx-pch     \
    --disable-multilib          \
    --disable-bootstrap         \
    --disable-libgomp           \
    --with-mpfr-include=$(pwd)/../gcc-4.7.2/mpfr/src \
    --with-mpfr-lib=$(pwd)/mpfr/src/.libs
```

The meaning of the new configure options:

--enable-clocale=gnu

This option ensures the correct locale model is selected for the C++ libraries under all circumstances. If the configure script finds the *de_DE* locale installed, it will select the correct gnu locale model. However, if the *de_DE* locale is not installed, there is the risk of building Application Binary Interface (ABI)-incompatible C++ libraries because the incorrect generic locale model may be selected.

--enable-threads=posix

This enables C++ exception handling for multi-threaded code.

--enable-__cxa_atexit

This option allows use of __cxa_atexit, rather than atexit, to register C++ destructors for local statics and global objects. This option is essential for fully standards-compliant handling of destructors. It also affects the C++ ABI, and therefore results in C++ shared libraries and C++ programs that are interoperable with other Linux distributions.

--enable-languages=c,c++
This option ensures that both the C and C++ compilers are built.

--disable-libstdcxx-pch
Do not build the pre-compiled header (PCH) for libstdc++. It takes up a lot of space, and we have no use for it.

--disable-bootstrap
For native builds of GCC, the default is to do a "bootstrap" build. This does not just compile GCC, but compiles it several times. It uses the programs compiled in a first round to compile itself a second time, and then again a third time. The second and third iterations are compared to make sure it can reproduce itself flawlessly. This also implies that it was compiled correctly. However, the LFS build method should provide a solid compiler without the need to bootstrap each time.

Compile the package:

```
make
```

Install the package:

```
make install
```

As a finishing touch, create a symlink. Many programs and scripts run **cc** instead of **gcc**, which is used to keep programs generic and therefore usable on all kinds of UNIX systems where the GNU C compiler is not always installed. Running **cc** leaves the system administrator free to decide which C compiler to install:

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

Caution

At this point, it is imperative to stop and ensure that the basic functions (compiling and linking) of the new toolchain are working as expected. To perform a sanity check, run the following commands:

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

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

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

Note that /tools/lib, or /tools/lib64 for 64-bit machines appears as the prefix of the dynamic linker.

If the output is not shown as above or there was no output at all, then something is wrong. Investigate and retrace the steps to find out where the problem is and correct it.

> This issue must be resolved before continuing on. First, perform the sanity check again, using **gcc** instead of **cc**. If this works, then the /tools/bin/cc symlink is missing. Install the symlink as per above. Next, ensure that the PATH is correct. This can be checked by running **echo $PATH** and verifying that /tools/bin is at the head of the list. If the PATH is wrong it could mean that you are not logged in as user *lfs* or that something went wrong back in Section 4.4, "Setting Up the Environment."
>
> Once all is well, clean up the test files:
>
> ```
> rm -v dummy.c a.out
> ```

Details on this package are located in Section 6.17.2, "Contents of GCC."

5.10. Tcl-8.6.0

The Tcl package contains the Tool Command Language.

Approximate build time:	0.4 SBU
Required disk space:	33 MB

5.10.1. Installation of Tcl

This package and the next three (Expect, DejaGNU, and Check) are installed to support running the test suites for GCC and Binutils and other packages. Installing four packages for testing purposes may seem excessive, but it is very reassuring, if not essential, to know that the most important tools are working properly. Even if the test suites are not run in this chapter (they are not mandatory), these packages are required to run the test suites in Chapter 6.

Prepare Tcl for compilation:

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

Build 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 Tcl test suite anyway, issue the following command:

```
TZ=UTC make test
```

The Tcl test suite may experience failures under certain host conditions that are not fully understood. Therefore, test suite failures here are not surprising, and are not considered critical. The *TZ=UTC* parameter sets the time zone to Coordinated Universal Time (UTC), also known as Greenwich Mean Time (GMT), but only for the duration of the test suite run. This ensures that the clock tests are exercised correctly. Details on the TZ environment variable are provided in Chapter 7.

Install the package:

```
make install
```

Make the installed library writable so debugging symbols can be removed later:

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

Install Tcl's headers. The next package, Expect, requires them to build.

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

Now make a necessary symbolic link:

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

5.10.2. Contents of Tcl

Installed programs: tclsh (link to tclsh8.6) and tclsh8.6
Installed library: libtcl8.6.so, libtclstub8.6.a

Short Descriptions

tclsh8.6	The Tcl command shell
tclsh	A link to tclsh8.6
libtcl8.6.so	The Tcl library
libtclstub8.6.a	The Tcl Stub library

5.11. Expect-5.45

The Expect package contains a program for carrying out scripted dialogues with other interactive programs.

Approximate build time: 0.1 SBU
Required disk space: 4.4 MB

5.11.1. Installation of Expect

First, force Expect's configure script to use /bin/stty instead of a /usr/local/bin/stty it may find on the host system. This will ensure that our test suite tools remain sane for the final builds of our toolchain:

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

Now prepare Expect for compilation:

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

The meaning of the configure options:

--with-tcl=/tools/lib
    This ensures that the configure script finds the Tcl installation in the temporary tools location

instead of possibly locating an existing one on the host system.

--with-tclinclude=/tools/include
This explicitly tells Expect where to find Tcl's internal headers. Using this option avoids conditions where **configure** fails because it cannot automatically discover the location of Tcl's headers.

Build 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 Expect test suite anyway, issue the following command:

```
make test
```

Note that the Expect test suite is known to experience failures under certain host conditions that are not within our control. Therefore, test suite failures here are not surprising and are not considered critical.

Install the package:

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

The meaning of the make parameter:

SCRIPTS=""
This prevents installation of the supplementary Expect scripts, which are not needed.

5.11.2. Contents of Expect

Installed program:	expect
Installed library:	libexpect-5.45.a

Short Descriptions

expect	Communicates with other interactive programs according to a script
libexpect-5.45.a	Contains functions that allow Expect to be used as a Tcl extension or to be used directly from C or C++ (without Tcl)

5.12. DejaGNU-1.5

The DejaGNU package contains a framework for testing other programs.

Approximate build time:	less than 0.1 SBU
Required disk space:	4.1 MB

5.12.1. Installation of DejaGNU

Prepare DejaGNU for compilation:

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

Build and install the package:

```
make install
```

To test the results, issue:

```
make check
```

5.12.2. Contents of DejaGNU

Installed program: runtest

Short Descriptions

runtest   A wrapper script that locates the proper **expect** shell and then runs
             DejaGNU

5.13. Check-0.9.9

Check is a unit testing framework for C.

Approximate build time: 0.1 SBU
Required disk space: 6.9 MB

5.13.1. Installation of Check

Prepare Check for compilation:

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

Build 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 Check test suite anyway, issue the following
command:

```
make check
```

Note that the Check test suite may take a relatively long (up to 4 SBU) time.

Install the package:

```
make install
```

5.13.2. Contents of Check

Installed program:	checkmk
Installed library:	libcheck.{a,so}

Short Descriptions

checkmk	Awk script for generating C unit tests for use with the Check unit testing framework
libcheck. {a,so}	Contains functions that allow Check to be called from a test program

5.14. Ncurses-5.9

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

Approximate build time:	0.5 SBU
Required disk space:	35 MB

5.14.1. Installation of Ncurses

Prepare Ncurses for compilation:

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

The meaning of the configure options:

--without-ada

This ensures that Ncurses does not build support for the Ada compiler which may be present on the host but will not be available once we enter the **chroot** environment.

--enable-overwrite

This tells Ncurses to install its header files into /tools/include, instead of /tools/include /ncurses, to ensure that other packages can find the Ncurses headers successfully.

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
```

Details on this package are located in Section 6.21.2, "Contents of Ncurses."

5.15. Bash-4.2

The Bash package contains the Bourne-Again SHell.

Approximate build time:	0.4 SBU
Required disk space:	48 MB

5.15.1. Installation of Bash

First, apply the following patch to fix various bugs that have been addressed upstream:

```
patch -Np1 -i ../bash-4.2-fixes-11.patch
```

Prepare Bash for compilation:

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

The meaning of the configure options:

--without-bash-malloc
> This option turns off the use of Bash's memory allocation (malloc) function which is known to cause segmentation faults. By turning this option off, Bash will use the malloc functions from Glibc which are more stable.

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 Bash test suite anyway, issue the following command:

```
make tests
```

Install the package:

```
make install
```

Make a link for the programs that use **sh** for a shell:

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

Details on this package are located in [Section 6.33.2, "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**.

Approximate build time:	less than 0.1 SBU
Required disk space:	5.7 MB

5.16.1. Installation of Bzip2

The Bzip2 package does not contain a `configure` script. Compile and test it with:

```
make
```

Install the package:

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

Details on this package are located in <u>Section 6.19.2, "Contents of Bzip2."</u>

5.17. Coreutils-8.21

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

Approximate build time:	0.8 SBU
Required disk space:	133 MB

5.17.1. Installation of Coreutils

Prepare Coreutils for compilation:

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

The meaning of the configure options:

`--enable-install-program=hostname`
This enables the `hostname` binary to be built and installed – it is disabled by default but is required by the Perl test suite.

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 Coreutils test suite anyway, issue the following command:

```
make RUN_EXPENSIVE_TESTS=yes check
```

The *RUN_EXPENSIVE_TESTS=yes* parameter tells the test suite to run several additional tests that are considered relatively expensive (in terms of CPU power and memory usage) on some platforms, but generally are not a problem on Linux.

Install the package:

```
make install
```

Details on this package are located in <u>Section 6.27.2, "Contents of Coreutils."</u>

5.18. Diffutils-3.2

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

Approximate build time:　　　　0.2 SBU
Required disk space:　　　　　8.5 MB

5.18.1. Installation of Diffutils

Fix an incompatibility between this package and Glibc-2.17:

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

Prepare Diffutils 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 Diffutils test suite anyway, issue the following command:

```
make check
```

Install the package:

```
make install
```

Details on this package are located in <u>Section 6.40.2, "Contents of Diffutils."</u>

5.19. File-5.13

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

Approximate build time:　　　　0.1 SBU
Required disk space:　　　　　12.4 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 <u>Section 6.12.2, "Contents of File."</u>

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).

Approximate build time:	0.2 SBU
Required disk space:	27 MB

5.20.1. Installation of Findutils

Prepare Findutils 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 Findutils test suite anyway, issue the following command:

```
make check
```

Install the package:

```
make install
```

Details on this package are located in <u>Section 6.42.2, "Contents of Findutils."</u>

5.21. Gawk-4.0.2

The Gawk package contains programs for manipulating text files.

Approximate build time:	0.2 SBU
Required disk space:	30 MB

5.21.1. Installation of Gawk

Prepare Gawk 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 Gawk test suite anyway, issue the following command:

```
make check
```

Install the package:

```
make install
```

Details on this package are located in <u>Section 6.41.2, "Contents of Gawk."</u>

5.22. Gettext-0.18.2

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.

Approximate build time:	0.6 SBU
Required disk space:	101 MB

5.22.1. Installation of Gettext

For our temporary set of tools, we only need to build and install one binary from Gettext.

Prepare Gettext for compilation:

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

The meaning of the configure option:

EMACS="no"
    This prevents the configure script from determining where to install Emacs Lisp files as the test is known to hang on some hosts.

--disable-shared
    We do not need to install any of the shared Gettext libraries at this time, therefore there is no need to build them.

Compile the package:

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

As only one binary has been compiled, it is not possible to run the test suite without compiling additional support libraries from the Gettext package. It is therefore not recommended to attempt to run the test suite at this stage.

Install the **msgfmt** binary:

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

Details on this package are located in <u>Section 6.44.2, "Contents of Gettext."</u>

5.23. Grep-2.14

The Grep package contains programs for searching through files.

| **Approximate build time:** | 0.2 SBU |
| **Required disk space:** | 21 MB |

5.23.1. Installation of Grep

Prepare Grep 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 Grep test suite anyway, issue the following command:

```
make check
```

Install the package:

```
make install
```

Details on this package are located in <u>Section 6.31.2, "Contents of Grep."</u>

5.24. Gzip-1.5

The Gzip package contains programs for compressing and decompressing files.

| **Approximate build time:** | 0.2 SBU |
| **Required disk space:** | 10 MB |

5.24.1. Installation of Gzip

Prepare Gzip 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 Gzip test suite anyway, issue the following command:

```
make check
```

Install the package:

```
make install
```

Details on this package are located in <u>Section 6.49.2, "Contents of Gzip."</u>

5.25. M4-1.4.16

The M4 package contains a macro processor.

Approximate build time:	0.2 SBU
Required disk space:	16.6 MB

5.25.1. Installation of M4

Fix an incompatibility between this package and Glibc-2.17:

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

Prepare M4 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 M4 test suite anyway, issue the following command:

```
make check
```

Install the package:

```
make install
```

Details on this package are located in <u>Section 6.29.2, "Contents of M4."</u>

5.26. Make-3.82

The Make package contains a program for compiling packages.

Approximate build time:	0.1 SBU
Required disk space:	11.2 MB

5.26.1. Installation of Make

Prepare Make 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 Make test suite anyway, issue the following command:

```
make check
```

Install the package:

```
make install
```

Details on this package are located in <u>Section 6.54.2, "Contents of Make."</u>

5.27. Patch-2.7.1

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

Approximate build time: 0.1 SBU
Required disk space: 3.4 MB

5.27.1. Installation of Patch

Prepare Patch 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 Patch test suite anyway, issue the following command:

```
make check
```

Install the package:

```
make install
```

Details on this package are located in <u>Section 6.56.2, "Contents of Patch."</u>

5.28. Perl-5.16.2

The Perl package contains the Practical Extraction and Report Language.

Approximate build time: 1.8 SBU
Required disk space: 237 MB

5.28.1. Installation of Perl

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

```
patch -Np1 -i ../perl-5.16.2-libc-1.patch
```

Prepare Perl for compilation:

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

Build the package:

```
make
```

Although Perl comes with a test suite, it would be better to wait until it is installed in the next chapter.

Only a few of the utilities and libraries, need to be installed at this time:

```
cp -v perl cpan/podlators/pod2man /tools/bin
mkdir -pv /tools/lib/perl5/5.16.2
cp -Rv lib/* /tools/lib/perl5/5.16.2
```

Details on this package are located in <u>Section 6.37.2, "Contents of Perl."</u>

5.29. Sed-4.2.2

The Sed package contains a stream editor.

Approximate build time:	0.1 SBU
Required disk space:	10.5 MB

5.29.1. Installation of Sed

Prepare Sed 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 Sed test suite anyway, issue the following command:

```
make check
```

Install the package:

```
make install
```

Details on this package are located in <u>Section 6.18.2, "Contents of Sed."</u>

5.30. Tar-1.26

The Tar package contains an archiving program.

Approximate build time:	0.4 SBU
Required disk space:	20.6 MB

5.30.1. Installation of Tar

Fix an incompatibility between this package and Glibc-2.17:

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

Prepare Tar 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 Tar test suite anyway, issue the following command:

```
make check
```

Install the package:

```
make install
```

Details on this package are located in Section 6.59.2, "Contents of Tar."

5.31. Texinfo-5.0

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

Approximate build time:	0.3 SBU
Required disk space:	94 MB

5.31.1. Installation of Texinfo

Prepare Texinfo 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 Texinfo test suite anyway, issue the following command:

```
make check
```

Install the package:

```
make install
```

Details on this package are located in <u>Section 6.60.2, "Contents of Texinfo."</u>

5.32. Xz-5.0.4

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.

Approximate build time:	0.2 SBU
Required disk space:	16.3 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 <u>Section 6.46.2, "Contents of Xz."</u>

5.33. Stripping

The steps in this section are optional, but if the LFS partition is rather small, it is beneficial to learn that unnecessary items can be removed. The executables and libraries built so far contain about 70 MB of unneeded debugging symbols. Remove those symbols with:

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

These commands will skip a number of files, reporting that it does not recognize their file format. Most of these are scripts instead of binaries.

Take care *not* to use `--strip-unneeded` on the libraries. The static ones would be destroyed and the toolchain packages would need to be built all over again.

To save more, remove the documentation:

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

At this point, you should have at least 850 MB of free space in `$LFS` that can be used to build and install Glibc in the next phase. If you can build and install Glibc, you can build and install the rest too.

5.34. Changing Ownership

Note

The commands in the remainder of this book must be performed while logged in as user *root* and no longer as user *lfs*. Also, double check that `$LFS` is set in *root*'s environment.

Currently, the `$LFS/tools` directory is owned by the user *lfs*, a user that exists only on the host system. If the `$LFS/tools` directory is kept as is, the files are owned by a user ID without a corresponding account. This is dangerous because a user account created later could get this same user ID and would own the `$LFS/tools` directory and all the files therein, thus exposing these files to possible malicious manipulation.

To avoid this issue, you could add the *lfs* user to the new LFS system later when creating the `/etc/passwd` file, taking care to assign it the same user and group IDs as on the host system. Better yet, change the ownership of the `$LFS/tools` directory to user *root* by running the following command:

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

Although the `$LFS/tools` directory can be deleted once the LFS system has been finished, it can be retained to build additional LFS systems *of the same book version*. How best to backup `$LFS/tools` is a matter of personal preference.

Caution

If you intend to keep the temporary tools for use in building future LFS systems, *now* is the time to back them up. Subsequent commands in chapter 6 will alter the tools currently in place, rendering them useless for future builds.

Part III. Building the LFS System

Chapter 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.

Note

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 proc proc $LFS/proc
mount -vt sysfs sysfs $LFS/sys
```

In some host systems, /dev/shm is a symbolic link to /run/shm. Inside a chroot environment, this temporary file system needs to be mounted separate from the host file system:

```
if [ -h $LFS/dev/shm ]; then
  link=$(readlink $LFS/dev/shm)
  mkdir -p $LFS/$link
  mount -vt tmpfs shm $LFS/$link
  unset link
else
  mount -vt tmpfs shm $LFS/dev/shm
fi
```

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/sysconfig/rc.site, /etc/sysconfig /network, and /etc/sysconfig/ifconfig.eth0.

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 Section 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.

Note

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 Section 6.2.2, "Mounting and Populating /dev" and Section 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,sysconfig},home,lib,mnt,opt,run}
mkdir -pv /{media/{floppy,cdrom},sbin,srv,var}
install -dv -m 0750 /root
install -dv -m 1777 /tmp /var/tmp
mkdir -pv /usr/{,local/}{bin,include,lib,sbin,src}
mkdir -pv /usr/{,local/}share/{doc,info,locale,man}
mkdir -v  /usr/{,local/}share/{misc,terminfo,zoneinfo}
mkdir -pv /usr/{,local/}share/man/man{1..8}
for dir in /usr /usr/local; do
  ln -sv share/{man,doc,info} $dir
done
case $(uname -m) in
 x86_64) ln -sv lib /lib64 && ln -sv lib /usr/lib64 ;;
esac
mkdir -v /var/{log,mail,spool}
ln -sv /run /var/run
```

```
ln -sv /run/lock /var/lock
mkdir -pv /var/{opt,cache,lib/{misc,locate},local}
```

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
sed 's/tools/usr/' /tools/lib/libstdc++.la > /usr/lib/libstdc++.la
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:
tape:x:4:
tty:x:5:
daemon:x:6:
floppy:x:7:
disk:x:8:
lp:x:9:
dialout:x:10:
audio:x:11:
video:x:12:
utmp:x:13:
usb:x:14:
cdrom:x:15:
mail:x:34:
nogroup:x:99:
EOF
```

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 Chapter 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/log/{btmp,lastlog,wtmp}
chgrp -v utmp /var/log/lastlog
chmod -v 664  /var/log/lastlog
chmod -v 600  /var/log/btmp
```

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.

> **Note**
>
> The /run/utmp file records the users that are currently logged in. This file is created
> dynamically in the boot scripts.

6.7. Linux-3.8.1 API Headers

The Linux API Headers (in linux-3.8.1.tar.xz) expose the kernel's API for use by Glibc.

Approximate build time:	0.1 SBU
Required disk space:	515 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-3.47

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

Approximate build time: less than 0.1 SBU
Required disk space: 22 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-2.17

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.

Approximate build time: 17.6 SBU
Required disk space: 852 MB

6.9.1. Installation of Glibc

Note

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.

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

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

Prepare Glibc for compilation:

```
../glibc-2.17/configure  \
    --prefix=/usr          \
    --disable-profile      \
    --enable-kernel=2.6.25 \
    --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
```

Important

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

Generally a few tests do not pass, but you can generally ignore any of the test failures listed below. Now test the build results:

```
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* and *conform/run-conformtest* tests. 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*, *tst/tst-cputimer1*, 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.

- 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**).

- Other tests known to fail on some architectures are posix/bug-regex32, misc/tst-writev, elf/check-textrel, nptl/tst-getpid2, and stdio-common/bug22.

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
```

Install NIS and RPC related headers that are not installed by default; these are required to rebuild glibc and by several BLFS packages:

```
cp -v ../glibc-2.17/sunrpc/rpc/*.h /usr/include/rpc
cp -v ../glibc-2.17/sunrpc/rpcsvc/*.h /usr/include/rpcsvc
cp -v ../glibc-2.17/nis/rpcsvc/*.h /usr/include/rpcsvc
```

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_GB -f UTF-8 en_GB.UTF-8
localedef -i en_HK -f ISO-8859-1 en_HK
localedef -i en_PH -f ISO-8859-1 en_PH
localedef -i en_US -f ISO-8859-1 en_US
localedef -i en_US -f UTF-8 en_US.UTF-8
localedef -i es_MX -f ISO-8859-1 es_MX
localedef -i fa_IR -f UTF-8 fa_IR
localedef -i fr_FR -f ISO-8859-1 fr_FR
localedef -i fr_FR@euro -f ISO-8859-15 fr_FR@euro
localedef -i fr_FR -f UTF-8 fr_FR.UTF-8
localedef -i it_IT -f ISO-8859-1 it_IT
localedef -i it_IT -f UTF-8 it_IT.UTF-8
localedef -i ja_JP -f EUC-JP ja_JP
localedef -i ru_RU -f KOI8-R ru_RU.KOI8-R
localedef -i ru_RU -f UTF-8 ru_RU.UTF-8
localedef -i tr_TR -f UTF-8 tr_TR.UTF-8
```

```
localedef -i zh_CN -f GB18030 zh_CN.GB18030
```

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

Alternatively, install all locales listed in the glibc-2.17/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-2.17/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
```

Install timezone data:

```
tar -xf ../tzdata2012j.tar.gz

ZONEINFO=/usr/share/zoneinfo
mkdir -pv $ZONEINFO/{posix,right}

for tz in etcetera southamerica northamerica europe africa antarctica  \
          asia australasia backward pacificnew solar87 solar88 solar89 \
          systemv; do
    zic -L /dev/null   -d $ZONEINFO       -y "sh yearistype.sh" ${tz}
    zic -L /dev/null   -d $ZONEINFO/posix -y "sh yearistype.sh" ${tz}
    zic -L leapseconds -d $ZONEINFO/right -y "sh yearistype.sh" ${tz}
done

cp -v zone.tab iso3166.tab $ZONEINFO
```

```
zic -d $ZONEINFO -p America/New_York
unset ZONEINFO
```

The meaning of the zic commands:

zic -L /dev/null ...
> This creates posix timezones, without any leap seconds. It is conventional to put these in both zoneinfo and zoneinfo/posix. It is necessary to put the POSIX timezones in zoneinfo, otherwise various test-suites will report errors. On an embedded system, where space is tight and you do not intend to ever update the timezones, you could save 1.9MB by not using the posix directory, but some applications or test-suites might give less good results

zic -L leapseconds ...
> This creates right timezones, including leap seconds. On an embedded system, where space is tight and you do not intend to ever update the timezones, or care about the correct time, you could save 1.9MB by omitting the right directory.

zic ... -p ...
> This creates the posixrules file. We use New York because POSIX requires the daylight savings time rules to be in accordance with US rules.

One way to determine the local time zone is to 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

EOF
```

If desired, the dynamic loader can also search a directory and include the contents of files found there. Generally the files in this include directory are one line specifying the desired library path. To add this capability run the following commands:

```
cat >> /etc/ld.so.conf << "EOF"
# Add an include directory
include /etc/ld.so.conf.d/*.conf

EOF
mkdir /etc/ld.so.conf.d
```

6.9.4. Contents of Glibc

Installed programs:	catchsegv, gencat, getconf, getent, iconv, iconvconfig, ldconfig, ldd, lddlibc4, locale, localedef, makedb, mtrace, nscd, pcprofiledump, pldd, pt_chown, rpcgen, sln, sotruss, 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/audit, /usr/lib/gconv, /usr/lib/glibc, /usr/lib/locale, /usr/share/i18n, /usr/share/zoneinfo, /var/db

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
makedb	Creates a simple database from textual input
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
pldd	Lists dynamic shared objects used by running processes
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
sln	A statically linked **ln** program
sotruss	Traces shared library procedure calls of a specified command
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-2.17/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. Adjusting the Toolchain

Now that the final C libraries have been installed, it is time to adjust the toolchain so that it will link any newly compiled program against these new libraries.

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-1.2.7

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

Approximate build time:	0.1 SBU
Required disk space:	3.9 MB

6.11.1. Installation of Zlib

Prepare Zlib for compilation:

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

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.1.2.7 /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. File-5.13

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

Approximate build time: 0.1 SBU
Required disk space: 12.5 MB

6.12.1. 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
```

6.12.2. 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

6.13. Binutils-2.23.1

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

Approximate build time: 1.9 SBU
Required disk space: 343 MB

6.13.1. 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
```

Fix some test suite scripts so all tests pass:

```
patch -Np1 -i ../binutils-2.23.1-testsuite_fix-1.patch
```

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-2.23.1/configure --prefix=/usr --enable-shared
```

Note

There is an optional argument to **configure**, `--enable-lto`, that can be used to allow the **ar**, **nm**, and **ranlib** commands to accept a `--plugin` parameter. This is used to allow **gcc** to do "link time optimization" if specified. No packages in LFS or BLFS currently use this capability.

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).

> ### Important
>
> 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-2.23.1/include/libiberty.h /usr/include
```

6.13.2. Contents of Binutils

Installed programs:	addr2line, ar, as, c++filt, elfedit, gprof, ld, ld.bfd, 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
elfedit	Updates the ELF header of ELF files
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
ld.bfd	Hard link to **ld**
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**.

6.14. GMP-5.1.1

The GMP package contains math libraries. These have useful functions for arbitrary precision arithmetic.

Approximate build time:	1.2 SBU
Required disk space:	50 MB

6.14.1. Installation of GMP

> **Note**
>
> 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
```

The meaning of the new configure options:

--enable-cxx
This parameter enables C++ support

Compile the package:

```
make
```

> **Important**
>
> 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 184 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-5.1.1
cp    -v doc/{isa_abi_headache,configuration} doc/*.html \
         /usr/share/doc/gmp-5.1.1
```

6.14.2. Contents of GMP

Installed Libraries:	libgmp.{a,so}, libgmpxx.{a,so}, and libmp.{a,so}
Installed directory:	/usr/share/doc/gmp-5.1.1

Short Descriptions

libgmp	Contains precision math functions.
libgmpxx	Contains C++ precision math functions.
libmp	Contains the Berkeley MP math functions.

6.15. MPFR-3.1.1

The MPFR package contains functions for multiple precision math.

Approximate build time:	0.8 SBU
Required disk space:	27 MB

6.15.1. Installation of MPFR

Prepare MPFR for compilation:

```
./configure  --prefix=/usr      \
             --enable-thread-safe \
```

```
                          --docdir=/usr/share/doc/mpfr-3.1.1
```

Compile the package:

```
make
```

> **Important**
>
> 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
```

6.15.2. Contents of MPFR

Installed Libraries:	libmpfr.{a,so}
Installed directory:	/usr/share/doc/mpfr-3.1.1

Short Descriptions

libmpfr   Contains multiple-precision math functions.

6.16. MPC-1.0.1

The MPC package contains a library for the arithmetic of complex numbers with arbitrarily high precision and correct rounding of the result.

Approximate build time:	0.4 SBU
Required disk space:	10.2 MB

6.16.1. 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
```

6.16.2. Contents of MPC

Installed Libraries: libmpc.{a,so}

Short Descriptions

    libmpc   Contains complex math functions

6.17. GCC-4.7.2

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

Approximate build time: 53.5 SBU
Required disk space: 2.0 GB

6.17.1. 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
```

Again, do not build the .info files. They are broken with the current version of **makeinfo**.

```
sed -i 's/BUILD_INFO=info/BUILD_INFO=/' gcc/configure
```

As in Section 5.9, "GCC-4.7.2 - Pass 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
```

Also fix an error in one of the check Makefiles:

```
sed -i -e /autogen/d -e /check.sh/d fixincludes/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-4.7.2/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.

Note

There is an optional argument to **configure**, --enable-lto, that can be used to allow **gcc** to do do "link time optimization" if specified. No packages in LFS or BLFS currently use this capability.

To use this feature, it must also be enabled in binutils.

Compile the package:

```
make
```

Important

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 32768
```

Test the results, but do not stop at errors:

```
make -k check
```

To receive a summary of the test suite results, run:

```
../gcc-4.7.2/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/7.3/** 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 to 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/4.7.2/../../../crt1.o succeeded
/usr/lib/gcc/i686-pc-linux-gnu/4.7.2/../../../crti.o succeeded
/usr/lib/gcc/i686-pc-linux-gnu/4.7.2/../../../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/lib/gcc/i686-pc-linux-gnu/4.7.2/include
 /usr/local/include
 /usr/lib/gcc/i686-pc-linux-gnu/4.7.2/include-fixed
 /usr/include
```

Again, note that the directory named after your target triplet may be different than the above, depending on your architecture.

Note

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
```

Finally, move a misplaced file:

```
mkdir -pv /usr/share/gdb/auto-load/usr/lib
mv -v /usr/lib/*gdb.py /usr/share/gdb/auto-load/usr/lib
```

6.17.2. Contents of GCC

Installed programs:	c++, cc (link to gcc), cpp, g++, gcc, gcc-ar, gcc-nm, gcc-ranlib, gccbug, and gcov
Installed libraries:	libgcc.a, libgcc_eh.a, libgcc_s.so, libgcov.a, libgomp.{a,so}, liblto_plugin.so, libmudflap.{a,so}, libmudflapth.{a,so}, libquadmath.{a,so}, libssp.{a,so}, libssp_nonshared.a, libstdc++.{a,so} and libsupc++.a
Installed directories:	/usr/include/c++, /usr/lib/gcc, /usr/share/gcc-4.7.2

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
gcc-ar	A wrapper around **ar** that adds a plugin to the command line. This program is only used to add "link time optization" and is not useful with the default build options.
gcc-nm	A wrapper around **nm** that adds a plugin to the command line. This program is only used to add "link time optization" and is not useful with the default build options.

gcc-ranlib	A wrapper around **ranlib** that adds a plugin to the command line. This program is only used to add "link time optization" and is not useful with the default build options.
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
liblto_plugin	GCC's Link Time Optimization (LTO) plugin allows GCC to perform optimizations across compilation units.
libmudflap	Contains routines that support GCC's bounds checking functionality
libquadmath	GCC Quad Precision Math Library API
libssp	Contains routines supporting GCC's stack-smashing protection functionality
libstdc++	The standard C++ library
libsupc++	Provides supporting routines for the C++ programming language

6.18. Sed-4.2.2

The Sed package contains a stream editor.

| **Approximate build time:** | 0.2 SBU |
| **Required disk space:** | 6.7 MB |

6.18.1. Installation of Sed

Prepare Sed for compilation:

```
./configure --prefix=/usr --bindir=/bin --htmldir=/usr/share/doc/sed-4.2.2
```

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
```

6.18.2. Contents of Sed

Installed program: sed
Installed directory: /usr/share/doc/sed-4.2.2

Short Descriptions

sed   Filters and transforms text files in a single pass

6.19. 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**.

Approximate build time: less than 0.1 SBU
Required disk space: 6.9 MB

6.19.1. 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
```

Ensure the man pages are installed into the correct location:

```
sed -i "s@(PREFIX)/man@(PREFIX)/share/man@g" 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
```

6.19.2. Contents of Bzip2

Installed programs:	bunzip2 (link to bzip2), bzcat (link to bzip2), bzcmp (link to bdiff), 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

6.20. Pkg-config-0.28

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.

Approximate build time: 0.4 SBU
Required disk space: 31 MB

6.20.1. Installation of Pkg-config

Prepare Pkg-config for compilation:

```
./configure --prefix=/usr          \
            --with-internal-glib   \
            --disable-host-tool    \
            --docdir=/usr/share/doc/pkg-config-0.28
```

The meaning of the new configure options:

--with-internal-glib
> This will allow pkg-config to use it's internal version of glib because an external version is not available in LFS.

--disable-host-tool
> This option disables the creation of an undesired hard link to the pkg-config program.

Compile the package:

```
make
```

To test the results, issue:

```
make check
```

Install the package:

```
make install
```

6.20.2. Contents of Pkg-config

Installed program: pkg-config
Installed directory: /usr/share/doc/pkg-config-0.28

Short Descriptions

pkg-config   returns meta information for the specified library or package.

6.21. Ncurses-5.9

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

Approximate build time:	0.6 SBU
Required disk space:	40 MB

6.21.1. Installation of Ncurses

Prepare Ncurses for compilation:

```
./configure --prefix=/usr          \
            --mandir=/usr/share/man \
            --with-shared           \
            --without-debug         \
            --enable-pc-files       \
            --enable-widec
```

The meaning of the configure option:

--enable-widec
> This switch causes wide-character libraries (e.g., libncursesw.so.5.9) to be built instead of normal ones (e.g., libncurses.so.5.9). 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.

--enable-pc-files
> This switch generates and installs .pc files for pkg-config.

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
    ln -sfv ${lib}w.pc        /usr/lib/pkgconfig/${lib}.pc
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-5.9
cp -v -R doc/* /usr/share/doc/ncurses-5.9
```

Note

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
```

6.21.2. Contents of Ncurses

Installed programs:	captoinfo (link to tic), clear, infocmp, infotocap (link to tic), ncursesw5-config, reset (link to tset), tabs, 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, /usr/share/doc/ncurses-5.9

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
tabs	Clears and sets tab stops on a terminal
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

6.22. Util-linux-2.22.2

The Util-linux package contains miscellaneous utility programs. Among them are utilities for handling file systems, consoles, partitions, and messages.

Approximate build time:	0.7 SBU
Required disk space:	83 MB

6.22.1. 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 -i -e 's@etc/adjtime@var/lib/hwclock/adjtime@g' \
     $(grep -rl '/etc/adjtime' .)

mkdir -pv /var/lib/hwclock
```

6.22.2. Installation of Util-linux

```
./configure --disable-su --disable-sulogin --disable-login
```

The meaning of the configure option:

--disable-*
These switches disable building su, sulogin, and login. They duplicate the same programs provided by Section 6.26, "Shadow-4.1.5.1" and Section 6.58, "Sysvinit-2.88dsf". They also require **Linux-PAM** which is not available in LFS.

Compile the package:

```
make
```

This package does not come with a test suite.

Install the package:

```
make install
```

6.22.3. Contents of Util-linux

Installed programs:	addpart, agetty, blkid, blockdev, cal, cfdisk, chcpu, chrt, col, colcrt, colrm, column, ctrlaltdel, cytune, delpart, dmesg, eject, fallocate, fdformat, fdisk, findfs, findmnt, flock, fsck, fsck.cramfs, fsck.minix, fsfreeze, fstrim, getopt, hexdump, hwclock, i386, ionice, ipcmk, ipcrm, ipcs, isosize, ldattach, linux32, linux64, logger, look, losetup, lsblk, lscpu, lslocks, mcookie, mkfs, mkfs.bfs, mkfs.cramfs, mkfs.minix, mkswap, more, mount, mountpoint, namei, partx, pg, pivot_root, prlimit, raw, readprofile, rename, renice, resizepart, rev, rtcwake, script, scriptreplay, setarch, setsid, setterm, sfdisk, swaplabel, swapoff (link to swapon), swapon, switch_root, tailf, taskset, tunelp, ul, umount, unshare, utmpdump, uuidd, uuidgen, wall, wdctl, whereis, wipefs, and x86_64
Installed libraries:	libblkid.{a,so}, libmount.{a,so}, libuuid.{a,so}
Installed directories:	/usr/include/blkid, /usr/include/libmount, /usr/include/uuid, /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
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
chcpu	Modifies the state of CPUs
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
delpart	Asks the Linux kernel to remove a partition

dmesg	Dumps the kernel boot messages
eject	Ejects removable media
fallocate	Preallocates space to a file
fdformat	Low-level formats a floppy disk
fdisk	Manipulates the parition 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
kill	Sends signals to processes
ldattach	Attaches a line discipline to a serial line
linux32	A symbolic link to setarch
linux64	A symbolic link to setarch
logger	Enters the given message into the system log
look	Displays lines that begin with the given string
losetup	Sets up and controls loop devices
lsblk	Lists information about all or selected block devices in a tree-like format.
lscpu	Prints CPU architecture information
lslocks	Lists local system locks
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
mountpoint	Checks if the directory is a mountpoint
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
prlimit	Get and set a process' resource limits
raw	Bind a Linux raw character device to a block device
readprofile	Reads kernel profiling information
rename	Renames the given files, replacing a given string with another
renice	Alters the priority of running processes
resizepart	Asks the Linux kernel to resize a partition
rev	Reverses the lines of a given file
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
utmpdump	Displays the content of the given login file in a more user-friendly format
uuidd	A daemon used by the UUID library to generate time-based UUIDs in a secure and guaranteed-unique fashion.
uuidgen	Creates new UUIDs. Each new UUID can reasonably be considered unique among all UUIDs created, on the local system and on other systems, in the past and in the future
wall	Displays the contents of a file or, by default, its standard input, on the terminals of all currently logged in users
wdctl	Shows hardware watchdog status
whereis	Reports the location of the binary, source, and man page for the given command
wipefs	Wipes a filesystem signature from a device
x86_64	A symbolic link to setarch
libblkid	Contains routines for device identification and token extraction
libmount	Contains routines for block device mounting and unmounting
libuuid	Contains routines for generating unique identifiers for objects that may be accessible beyond the local system

6.23. Psmisc-22.20

The Psmisc package contains programs for displaying information about running processes.

Approximate build time:	less than 0.1 SBU
Required disk space:	4.2 MB

6.23.1. 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
```

6.23.2. 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

6.24. Procps-ng-3.3.6

The Procps-ng package contains programs for monitoring processes.

Approximate build time:	0.2 SBU
Required disk space:	13 MB

6.24.1. Installation of Procps-ng

Now prepare procps-ng for compilation:

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

```
                        --libdir=/usr/lib                        \
                        --docdir=/usr/share/doc/procps-ng-3.3.6 \
                        --disable-static                         \
                        --disable-skill                          \
                        --disable-kill
```

The meaning of the configure options:

--disable-skill

    This switch disables the obsolete and unportable skill and snice commands.

--disable-kill

    This switch disables building the kill command that was installed in the util-linux package.

Compile the package:

```
make
```

The test suite needs some custom modifications for LFS. The **which** command is not available, the **pmap** test does not match a newline character in two tests, and the **slabtop** test may return more than 999,999 objects. To run the test suite, run the following commands:

```
pushd testsuite
  sed -i -e 's|exec which sleep|exec echo /tools/bin/sleep|' \
         -e 's|999999|&9|'                        config/unix.exp
  sed -i -e 's|pmap_initname\\\$|pmap_initname|' pmap.test/pmap.exp
  make site.exp
  DEJAGNU=global-conf.exp runtest
popd
```

Install the package:

```
make install
```

Finally move the library to a location that can be found if /usr is not mounted.

```
mv -v /usr/lib/libprocps.so.* /lib
ln -sfv ../../lib/libprocps.so.1.1.0 /usr/lib/libprocps.so
```

6.24.2. Contents of Procps-ng

Installed programs: free, pgrep, pkill, pmap, ps, pwdx, slabtop, sysctl, tload, top, uptime, vmstat, w, and, watch

Installed library: libprocps.so

Short Descriptions

free	Reports the amount of free and used memory (both physical and swap memory) in the system
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
slabtop	Displays detailed kernel slap cache information in real time
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

6.25. E2fsprogs-1.42.7

The E2fsprogs package contains the utilities for handling the *ext2* file system. It also supports the *ext3* and *ext4* journaling file systems.

| **Approximate build time:** | 1.7 SBU |
| **Required disk space:** | 64 MB |

6.25.1. 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, be sure to enable sufficient swap space for the test. See Section 2.3, "Creating a File System on the Partition" and Section 2.4, "Mounting the New Partition" 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
```

6.25.2. Contents of E2fsprogs

Installed programs:	badblocks, chattr, compile_et, debugfs, dumpe2fs, e2freefrag, e2fsck, e2image, e2initrd_helper, e2label, e2undo, e4defrag, filefrag, fsck.ext2, fsck.ext3, fsck.ext4, fsck.ext4dev, logsave, lsattr, mk_cmds, mke2fs, mkfs.ext2, mkfs.ext3, mkfs.ext4, mkfs.ext4dev, mklost+found, resize2fs, and tune2fs
Installed libraries:	libcom_err.{a,so}, libe2p.{a,so}, libext2fs.{a,so}, libquota.a and libss.{a,so}

Installed directory: /usr/include/e2p, /usr/include/et, /usr/include/ext2fs, /usr/include/quota, /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.
e4defrag	Online defragmenter for ext4 filesystems
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
libquota	Provides an interface for creating and updating quota files and ext4 superblock fields
libss	Used by **debugfs**

6.26. Shadow-4.1.5.1

The Shadow package contains programs for handling passwords in a secure way.

| **Approximate build time:** | 0.2 SBU |
| **Required disk space:** | 42 MB |

6.26.1. Installation of Shadow

Note

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 / /' {} \;
```

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
```

Note

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
```

6.26.2. 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 undesirable, 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
```

6.26.3. Setting the root password

Choose a password for user *root* and set it by running:

```
passwd root
```

6.26.4. 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

6.27. Coreutils-8.21

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

Approximate build time:	3.4 SBU
Required disk space:	116 MB

6.27.1. Installation of Coreutils

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-8.21-i18n-1.patch
```

> **Note**
>
> 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:

```
FORCE_UNSAFE_CONFIGURE=1 ./configure \
            --prefix=/usr           \
            --libexecdir=/usr/lib \
            --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. Make sure the PATH in the **su** environment includes /tools/bin.

```
su nobody -s /bin/bash \
            -c "PATH=$PATH 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,test,[} /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
```

6.27.2. 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, realpath, rm, rmdir, runcon, seq, sha1sum, sha224sum, sha256sum, sha384sum, sha512sum, shred, shuf, sleep, sort, split, stat, stdbuf, stty, sum, sync, tac, tail, tee, test, timeout, touch, tr, true, truncate, tsort, tty, uname, unexpand, uniq, unlink, users, vdir, wc, who, whoami, and yes
Installed library:	libstdbuf.so
Installed directory:	/usr/libexec/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
realpath	Prints the resolved path
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.so	Library used by **stdbuf**

6.28. Iana-Etc-2.30

The Iana-Etc package provides data for network services and protocols.

Approximate build time:	less than 0.1 SBU
Required disk space:	2.2 MB

6.28.1. 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
```

6.28.2. 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

6.29. M4-1.4.16

The M4 package contains a macro processor.

Approximate build time:	0.4 SBU
Required disk space:	26.6 MB

6.29.1. Installation of M4

Fix an incompatibility between this package and Glibc-2.17:

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

Prepare M4 for compilation:

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

Compile the package:

```
make
```

To test the results, first fix a test program and then run the test programs:

```
sed -i -e '41s/ENOENT/& || errno == EINVAL/' tests/test-readlink.h
make check
```

Install the package:

```
make install
```

6.29.2. 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.

6.30. Bison-2.7

The Bison package contains a parser generator.

Approximate build time: 1.3 SBU
Required disk space: 34 MB

6.30.1. 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
```

6.30.2. 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

6.31. Grep-2.14

The Grep package contains programs for searching through files.

Approximate build time:	0.4 SBU
Required disk space:	30 MB

6.31.1. 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
```

6.31.2. 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

6.32. Readline-6.2

The Readline package is a set of libraries that offers command-line editing and history capabilities.

Approximate build time:	0.1 SBU
Required disk space:	17.2 MB

6.32.1. 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
```

Apply a patch to fix a known bug that has been fixed upstream:

```
patch -Np1 -i ../readline-6.2-fixes-1.patch
```

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
```

6.32.2. 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

6.33. Bash-4.2

The Bash package contains the Bourne-Again SHell.

Approximate build time:	1.7 SBU
Required disk space:	45 MB

6.33.1. Installation of Bash

First, apply the following patch to fix various bugs that have been addressed upstream:

```
patch -Np1 -i ../bash-4.2-fixes-11.patch
```

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 nobody -s /bin/bash -c "PATH=$PATH 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
```

Note

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.

6.33.2. 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

6.34. Libtool-2.4.2

The Libtool package contains the GNU generic library support script. It wraps the complexity of using shared libraries in a consistent, portable interface.

Approximate build time:	3.0 SBU
Required disk space:	37 MB

6.34.1. 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
```

6.34.2. 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

6.35. GDBM-1.10

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.

Approximate build time:	0.1 SBU
Required disk space:	8.5 MB

6.35.1. Installation of GDBM

Prepare GDBM for compilation:

```
./configure --prefix=/usr --enable-libgdbm-compat
```

The meaning of the configure option:

--enable-libgdbm-compat
This switch enables the libgdbm compatibility library to be built, as some packages outside of LFS may require the older DBM routines it provides.

Compile the package:

```
make
```

To test the results, issue:

```
make check
```

Install the package:

```
make install
```

6.35.2. Contents of GDBM

Installed program: testgdbm
Installed libraries: libgdbm.{so,a} and libgdbm_compat.{so,a}

Short Descriptions

testgdbm	Tests and modifies a GDBM database
libgdbm	Contains functions to manipulate a hashed database

6.36. Inetutils-1.9.1

The Inetutils package contains programs for basic networking.

Approximate build time: 0.4 SBU
Required disk space: 27 MB

6.36.1. Installation of Inetutils

Fix an incompatibility between this package and Glibc-2.17

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

Prepare Inetutils for compilation:

```
./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
```

Move some programs so they are available if /usr is not accessible:

```
mv -v /usr/bin/{hostname,ping,ping6,traceroute} /bin
```

6.36.2. 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

6.37. Perl-5.16.2

The Perl package contains the Practical Extraction and Report Language.

| **Approximate build time:** | 7.5 SBU |
| **Required disk space:** | 247 MB |

6.37.1. 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 -k test
```

Install the package:

```
make install
```

6.37.2. Contents of Perl

Installed programs:	a2p, c2ph, config_data, corelist, cpan, cpan2dist, cpanp, cpanp-run-perl, enc2xs, find2perl, h2ph, h2xs, instmodsh, json_pp, libnetcfg, perl, perl5.16.2 (link to perl), perlbug, perldoc, perlivp, perlthanks (link to perlbug), piconv, pl2pm, pod2html, pod2latex, pod2man, pod2text, pod2usage, podchecker, podselect, prove, psed (link to s2p), pstruct (link to c2ph), ptar, ptardiff, ptargrep, s2p, shasum, splain, xsubpp, and zipdetails
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
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
json_pp	Converts data between certain input and output formats
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

perl5.16.2	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
ptargrep	A Perl program that applies pattern matching to the contents of files in a tar archive
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
zipdetails	Displays details about the internal structure of a Zip file

6.38. Autoconf-2.69

The Autoconf package contains programs for producing shell scripts that can automatically configure source code.

| **Approximate build time:** | 4.5 SBU |
| **Required disk space:** | 17.1 MB |

6.38.1. 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
```

6.38.2. 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**

6.39. Automake-1.13.1

The Automake package contains programs for generating Makefiles for use with Autoconf.

Approximate build time:	less than 0.1 SBU (34.1 SBU with tests)
Required disk space:	100 MB

6.39.1. Installation of Automake

Prepare Automake for compilation:

```
./configure --prefix=/usr --docdir=/usr/share/doc/automake-1.13.1
```

Compile the package:

```
make
```

Note

The tests take a very long time: over 30 SBUs. Running the tests is not recommended.

To test the results, issue:

```
make check
```

Install the package:

```
make install
```

6.39.2. Contents of Automake

Installed programs:	acinstall, aclocal, aclocal-1.13, automake, automake-1.13, 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-1.13, /usr/share/automake-1.13, /usr/share/doc/automake-1.13.1

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-1.13	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-1.13	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**

6.40. Diffutils-3.2

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

Approximate build time: 0.5 SBU
Required disk space: 25 MB

6.40.1. Installation of Diffutils

Fix an incompatibility between this package and Glibc-2.17

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

Prepare Diffutils for compilation:

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

Compile the package:

```
make
```

To test the results, issue:

```
make check
```

Install the package:

```
make install
```

6.40.2. 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

6.41. Gawk-4.0.2

The Gawk package contains programs for manipulating text files.

Approximate build time:	0.2 SBU
Required disk space:	30 MB

6.41.1. 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-4.0.2
cp    -v doc/{awkforai.txt,*.{eps,pdf,jpg}} /usr/share/doc/gawk-4.0.2
```

6.41.2. Contents of Gawk

Installed programs:	awk (link to gawk), dgawk, gawk, gawk-4.0.2, grcat, igawk, pgawk, pgawk-4.0.2, and pwcat
Installed directories:	/usr/lib/awk, /usr/share/awk

Short Descriptions

awk	A link to **gawk**

dgawk	An awk debugger
gawk	A program for manipulating text files; it is the GNU implementation of **awk**
gawk-4.0.2	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-4.0.2	Hard link to **pgawk**
pwcat	Dumps the password database /etc/passwd

6.42. 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).

Approximate build time:	0.4 SBU
Required disk space:	29 MB

6.42.1. 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
```

6.42.2. 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

6.43. Flex-2.5.37

The Flex package contains a utility for generating programs that recognize patterns in text.

Approximate build time: 0.4 SBU
Required disk space: 39 MB

6.43.1. Installation of Flex

First, fix some regression tests:

```
patch -Np1 -i ../flex-2.5.37-bison-2.6.1-1.patch
```

Prepare Flex for compilation:

```
./configure --prefix=/usr           \
            --docdir=/usr/share/doc/flex-2.5.37
```

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
```

6.43.2. Contents of Flex

Installed programs:	flex, flex++ (link to flex), and lex
Installed libraries:	libfl.a and libfl_pic.a
Installed directories:	/usr/share/doc/flex-2.5.37

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
flex++	An extension of flex, is used for generating C++ code and classes. It is a symbolic link to **flex**
lex	A script that runs **flex** in **lex** emulation mode
libfl.a	The flex library

6.44. Gettext-0.18.2

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.

Approximate build time:	2.3 SBU
Required disk space:	180 MB

6.44.1. Installation of Gettext

Prepare Gettext for compilation:

```
./configure --prefix=/usr \
            --docdir=/usr/share/doc/gettext-0.18.2
```

Compile the package:

```
make
```

To test the results (this takes a long time, around 3 SBUs), issue:

```
make check
```

Install the package:

```
make install
```

6.44.2. 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-0.18.2, /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.

6.45. Groff-1.22.2

The Groff package contains programs for processing and formatting text.

Approximate build time:	0.5 SBU
Required disk space:	83 MB

6.45.1. 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:

```
mkdir -p /usr/share/doc/groff-1.22/pdf
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
```

6.45.2. 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-1.22.2, /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

6.46. Xz-5.0.4

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.

Approximate build time:	0.3 SBU
Required disk space:	18 MB

6.46.1. Installation of Xz

Prepare Xz for compilation with:

```
./configure --prefix=/usr --libdir=/lib --docdir=/usr/share/doc/xz-5.0.4
```

Compile the package:

```
make
```

To test the results, issue:

```
make check
```

Install the package:

```
make pkgconfigdir=/usr/lib/pkgconfig install
```

6.46.2. Contents of Xz

Installed programs:	lzcat (link to xz), lzcmp (link to xzdiff), lzdiff (link to xzdiff), lzegrep (link to xzgrep), lzfgrep (link to xzgrep), lzgrep (link to xzgrep), lzless (link to xzless), lzma (link to xz), lzmadec, lzmainfo, lzmore (link to xzmore), unlzma (link to xz), unxz, (link to xz), xz, xzcat (link to xz), xzcmp (link to xzdiff), xzdec, xzdiff, xzegrep (link to xzgrep), xzfgrep (link to xzgrep), xzgrep, xzless, xzmore
Installed libraries:	liblzma.{a,so}
Installed directories:	/usr/include/lzma and /usr/share/doc/xz-5.0.4

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

6.47. GRUB-2.00

The GRUB package contains the GRand Unified Bootloader.

Approximate build time:	0.7 SBU
Required disk space:	112 MB

6.47.1. Installation of GRUB

Fix an incompatibility between this package and Glibc-2.17:

```
sed -i -e '/gets is a/d' grub-core/gnulib/stdio.in.h
```

Prepare GRUB for compilation:

```
./configure --prefix=/usr          \
            --sysconfdir=/etc       \
            --disable-grub-emu-usb \
            --disable-efiemu        \
            --disable-werror
```

The --disable-werror option allows the build to complete with warnings introduced by more recent flex versions. The other --disable switches minimize what is built by disabling features and testing programs not 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 Section 8.4, "Using GRUB to Set Up the Boot Process".

6.47.2. Contents of GRUB

Installed programs:	grub-bios-setup, grub-editenv, grub-fstest, grub-install, grub-kbdcomp, grub-menulst2cfg, grub-mkconfig, grub-mkimage, grub-mklayout, grub-mknetdir, grub-mkpasswd-pbkdf2, grub-mkrelpath, grub-mkrescue, grub-mkstandalone, grub-ofpathname, grub-probe, grub-reboot, grub-script-check, grub-set-default, grub-sparc64-setup
Installed directories:	/usr/lib/grub, /etc/grub.d, /usr/share/grub, /boot/grub

Short Descriptions

grub-bios-setup	Is a helper program for grub-install
grub-editenv	A tool to edit the environment block
grub-fstest	Tool to debug the filesystem driver
grub-install	Install GRUB on your drive
grub-kbdcomp	Script that converts an xkb layout into one recognized by GRUB
grub-menulst2cfg	Converts a GRUB Legacy menu.lst into a grub.cfg for use with GRUB 2
grub-mkconfig	Generate a grub config file
grub-mkimage	Make a bootable image of GRUB
grub-mklayout	Generates a GRUB keyboard layout file
grub-mknetdir	Prepares a GRUB netboot directory
grub-mkpasswd-pbkdf2	Generates an encrypted PBKDF2 password for use in the boot menu
grub-mkrelpath	Makes a system pathname relative to its root

grub-mkrescue	Make a bootable image of GRUB suitable for a floppy disk or CDROM/DVD
grub-mkstandalone	Generates a standalone image
grub-ofpathname	Is a helper program that prints the path of a GRUB device
grub-probe	Probe device information for a given path or device
grub-reboot	Sets the default boot entry for GRUB for the next boot only
grub-script-check	Checks GRUB configuration script for syntax errors
grub-set-default	Sets the default boot entry for GRUB
grub-sparc64-setup	Is a helper program for grub-setup

6.48. Less-451

The Less package contains a text file viewer.

Approximate build time:	less than 0.1 SBU
Required disk space:	3.8 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. Gzip-1.5

The Gzip package contains programs for compressing and decompressing files.

Approximate build time:	0.2 SBU
Required disk space:	19.7 MB

6.49.1. 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
```

6.49.2. 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

6.50. IPRoute2-3.8.0

The IPRoute2 package contains programs for basic and advanced IPV4-based networking.

| **Approximate build time:** | 0.1 SBU |
| **Required disk space:** | 7.3 MB |

6.50.1. 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 commands 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
sed -i /ARPD/d Makefile
sed -i 's/arpd.8//' man/man8/Makefile
```

Remove a compiler flag that causes the compilation to fail:

```
sed -i 's/-Werror//' Makefile
```

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=              \
    MANDIR=/usr/share/man \
    DOCDIR=/usr/share/doc/iproute2-3.8.0 install
```

6.50.2. Contents of IPRoute2

Installed programs:	bridge, 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/iproute2-3.8.0, /usr/lib/tc

Short Descriptions

bridge	Configures network bridges
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 continuously monitor the state of devices, addresses and routes
lnstat	Provides Linux network statistics. It is a generalized and more feature-complete replacement for the old **rtstat** program
nstat	Shows network statistics
routef	A component of **ip route**. This is for flushing the routing tables
routel	A component of **ip route**. This is for listing the routing tables
rtacct	Displays the contents of /proc/net/rt_acct
rtmon	Route monitoring utility
rtpr	Converts the output of **ip -o** back into a readable form
rtstat	Route status utility

ss	Similar to the **netstat** command; shows active connections
tc	Traffic Controlling Executable; this is for Quality Of Service (QOS) and Class Of Service (COS) implementations

tc qdisc allows users to setup the queueing discipline

tc class allows users to setup classes based on the queuing discipline scheduling

tc estimator allows users to estimate the network flow into a network

tc filter allows users to setup the QOS/COS packet filtering

tc policy allows users to setup the QOS/COS policies

6.51. Kbd-1.15.5

The Kbd package contains key-table files, console fonts, and keyboard utilities.

Approximate build time:	0.1 SBU
Required disk space:	20 MB

6.51.1. 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-1.15.5-backspace-1.patch
```

After patching, the Backspace key generates the character with code 127, and the Delete key generates a well-known escape sequence.

Fix a bug that causes some keymaps not to be loaded correctly:

```
sed -i -e '326 s/if/while/' src/loadkeys.analyze.l
```

Remove the redundant **resizecons** program (it requires the defunct svgalib to provide the video mode files - for normal use **setfont** sizes the console appropriately) together with its manpage.

```
sed -i 's/\(RESIZECONS_PROGS=\)yes/\1no/g' configure
sed -i 's/resizecons.8 //' man/man8/Makefile.in
```

Prepare Kbd for compilation:

```
./configure --prefix=/usr --datadir=/lib/kbd \
  --disable-vlock
```

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.

--disable-vlock
This option prevents the vlock utility from being built, as it requires the PAM library, which isn't available in the chroot environment.

Compile the package:

```
make
```

This package does not come with a test suite.

Install the package:

```
make install
```

Note

For some languages (e.g., Belarusian) the Kbd package doesn't provide a useful keymap where the stock "by" keymap assumes the ISO-8859-5 encoding, and the CP1251 keymap is normally used. Users of such languages have to download working keymaps separately.

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-1.15.5
cp -R -v doc/* \
        /usr/share/doc/kbd-1.15.5
```

6.51.2. Contents of Kbd

Installed programs:	chvt, deallocvt, dumpkeys, fgconsole, getkeycodes, kbdinfo, kbd_mode, kbdrate, loadkeys, loadunimap, mapscrn, openvt, psfaddtable (link to psfxtable), psfgettable (link to psfxtable), psfstriptable (link to psfxtable), psfxtable, setfont, setkeycodes, setleds, setmetamode, 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
kbdinfo	Obtains information about the status of a console
kbd_mode	Reports or sets the keyboard mode
kbdrate	Sets the keyboard repeat and delay rates

loadkeys	Loads the keyboard translation tables
loadunimap	Loads the kernel unicode-to-font mapping table
mapscrn	An obsolete program that used to load a user-defined output character mapping table into the console driver; this is now done by **setfont**
openvt	Starts a program on a new virtual terminal (VT)
psfaddtable	A link to **psfxtable**
psfgettable	A link to **psfxtable**
psfstriptable	A link to **psfxtable**
psfxtable	Handle Unicode character tables for console fonts
setfont	Changes the Enhanced Graphic Adapter (EGA) and Video Graphics Array (VGA) fonts on the console
setkeycodes	Loads kernel scancode-to-keycode mapping table entries; this is useful if there are unusual keys on the keyboard
setleds	Sets the keyboard flags and Light Emitting Diodes (LEDs)
setmetamode	Defines the keyboard meta-key handling
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

6.52. Kmod-12

The Kmod package contains libraries and utilities for loading kernel modules

Approximate build time:	0.1 SBU
Required disk space:	30 MB

6.52.1. Installation of Kmod

Prepare Kmod for compilation:

```
./configure --prefix=/usr         \
            --bindir=/bin         \
            --libdir=/lib         \
            --sysconfdir=/etc     \
            --disable-manpages    \
            --with-xz             \
            --with-zlib
```

The meaning of the configure options:

--with-*
These options enable Kmod to handle compressed kernel modules.

--disable-manpages

> This option prevents the man pages from being built, as they rely on libxslt, which isn't available in the chroot environment.

Compile the package:

```
make
```

To test the results, issue:

```
make check
```

Install the package, and create symlinks for compatibility with Module-Init-Tools, the package that previously handled Linux kernel modules:

```
make pkgconfigdir=/usr/lib/pkgconfig install

for target in depmod insmod modinfo modprobe rmmod; do
  ln -sv ../bin/kmod /sbin/$target
done

ln -sv kmod /bin/lsmod
```

6.52.2. Contents of Kmod

Installed programs: depmod (link to kmod), insmod (link to kmod), kmod, kmod-nolib, lsmod (link to kmod), modinfo (link to kmod), modprobe (link to kmod), and rmmod (link to kmod)

Installed libraries: /lib/kmod.so

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
kmod	Loads and unloads kernel modules
libkmod	This library is used by other programs to load and unload kernel modules
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. Libpipeline-1.2.2

The Libpipeline package contains a library for manipulating pipelines of subprocesses in a flexible and convenient way.

Approximate build time:	0.2 SBU
Required disk space:	7.4 MB

6.53.1. Installation of Libpipeline

Prepare Libpipeline for compilation:

```
PKG_CONFIG_PATH=/tools/lib/pkgconfig ./configure --prefix=/usr
```

The meaning of the configure options:

PKG_CONFIG_PATH
Use pkg-config to obtain the location of the test library metadata built in Section 5.13, "Check-0.9.9".

Compile the package:

```
make
```

To test the results, issue:

```
make check
```

Install the package:

```
make install
```

6.53.2. Contents of Libpipeline

Installed libraries: libpipeline.so

Short Descriptions

libpipeline   This library is used to safely construct pipelines between subprocesses

6.54. Make-3.82

The Make package contains a program for compiling packages.

Approximate build time:	0.4 SBU
Required disk space:	11.3 MB

6.54.1. Installation of Make

First apply some upstream patches:

```
patch -Np1 -i ../make-3.82-upstream_fixes-3.patch
```

Prepare Make for compilation:

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

Compile the package:

```
make
```

To test the results, issue:

```
make check
```

Install the package:

```
make install
```

6.54.2. 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

6.55. Man-DB-2.6.3

The Man-DB package contains programs for finding and viewing man pages.

Approximate build time:	0.5 SBU
Required disk space:	27 MB

6.55.1. Installation of Man-DB

Prepare Man-DB for compilation:

```
./configure --prefix=/usr                         \
            --libexecdir=/usr/lib                 \
            --docdir=/usr/share/doc/man-db-2.6.3  \
            --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 check
```

Install the package:

```
make install
```

6.55.2. 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.

Table 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

Language (code)	Encoding	Language (code)	Encoding
Bulgarian (bg)	CP1251	Traditional Chinese, Hong Kong (zh_HK)	BIG5HKSCS
Czech (cs)	ISO-8859-2	Traditional Chinese (zh_TW)	BIG5
Greek (el)	ISO-8859-7		

Note

Manual pages in languages not in the list are not supported.

6.55.3. Contents of Man-DB

Installed programs: accessdb, apropos (link to whatis), catman, lexgrog, man, mandb, manpath, whatis, and zsoelim

Installed libraries: libman.so, libmandb.so

Installed directories: /usr/lib/man-db, /usr/share/doc/man-db-2.6.3

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*
libman	Contains run-time support for **man**
libmandb	Contains run-time support for **man**

6.56. Patch-2.7.1

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

Approximate build time:	less than 0.1 SBU
Required disk space:	3.4 MB

6.56.1. Installation of Patch

Prepare Patch for compilation:

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

Compile the package:

```
make
```

To test the results, issue:

```
make check
```

Install the package:

```
make install
```

6.56.2. 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.

6.57. Sysklogd-1.5

The Sysklogd package contains programs for logging system messages, such as those given by the
kernel when unusual things happen.

Approximate build time:	less than 0.1 SBU
Required disk space:	0.6 MB

6.57.1. Installation of Sysklogd

Compile the package:

```
make
```

This package does not come with a test suite.

Install the package:

```
make BINDIR=/sbin install
```

6.57.2. 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
```

6.57.3. 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

6.58. Sysvinit-2.88dsf

The Sysvinit package contains programs for controlling the startup, running, and shutdown of the system.

Approximate build time:　　　　less than 0.1 SBU
Required disk space:　　　　1.4 MB

6.58.1. 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
```

Maintained versions of the **wall**, **mountpoint**, and **utmpdump** programs were installed earlier by Util-linux. Suppress the installation of Sysvinit's versions of these programs and their man pages:

```
sed -i -e '/utmpdump/d' \
       -e '/mountpoint/d' src/Makefile
```

Compile the package:

```
make -C src
```

This package does not come with a test suite.

Install the package:

```
make -C src install
```

6.58.2. Contents of Sysvinit

Installed programs:	bootlogd, fstab-decode, halt, init, killall5, last, lastb (link to last), mesg, pidof (link to killall5), poweroff (link to halt), reboot (link to halt), runlevel, shutdown, sulogin, and telinit (link to init)

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
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

6.59. Tar-1.26

The Tar package contains an archiving program.

Approximate build time: 2.4 SBU
Required disk space: 34 MB

6.59.1. Installation of Tar

Fix an incompatibility between this package and Glibc-2.17:

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

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-1.26
```

6.59.2. Contents of Tar

Installed programs: rmt and tar
Installed directory: /usr/share/doc/tar-1.26

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 |

6.60. Texinfo-5.0

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

Approximate build time: 0.6 SBU
Required disk space: 101 MB

6.60.1. 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
```

6.60.2. Contents of Texinfo

Installed programs:	info, infokey, install-info, makeinfo, pdftexi2dvi, pod2texi, texi2any, 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
`pod2texi`	Converts Pod to Texinfo format
`texi2any`	Translate Texinfo source documentation to various other formats
`texi2dvi`	Used to format the given Texinfo document into a device-independent file that can be printed
`texi2pdf`	Used to format the given Texinfo document into a Portable Document Format (PDF) file
`texindex`	Used to sort Texinfo index files

6.61. Udev-197 (Extracted from systemd-197)

The Udev package contains programs for dynamic creation of device nodes. The development of udev has been merged with systemd, but most of systemd is incompatible with LFS. Here we build and install just the needed udev files.

Approximate build time:	0.1 SBU
Required disk space:	23 MB

6.61.1. Installation of Udev

Note

This package is a little different from other packages. The initial package that is extracted is systemd-197.tar.xz even though the application we are installing is udev. After changing to the systemd directory, follow the instructions below.

The udev-lfs tarball contains LFS-specific files used to build Udev. Unpack it into the systemd source directory:

```
tar -xvf ../udev-lfs-197-2.tar.bz2
```

Compile the package:

```
make -f udev-lfs-197-2/Makefile.lfs
```

Install the package:

```
make -f udev-lfs-197-2/Makefile.lfs install
```

> ### Caution
>
> There are several places within the systemd source code that have explicit directory
> paths embedded. For instance, the binary version of the hardware database's path and file
> name used at run time, /etc/udev/hwdb.bin, cannot be changed without explicit
> changes to the source code.

Now initialize the hardware database:

```
build/udevadm hwdb --update
```

Finally set up the persistent network udev rules. This task will be explained in detail in Section 7.2.1, "Creating stable names for network interfaces". Note that the /sys and /proc filesystems must be mounted in the chroot environment as explained at the beginning of this chapter for the following script to work.

```
bash udev-lfs-197-2/init-net-rules.sh
```

6.61.2. Contents of Udev

Installed programs:	accelerometer, ata_id, cdrom_id, collect, mtd_probe, scsi_id, v4l_id, udevadm, and udevd
Installed libraries:	libudev.so
Installed directories:	/etc/udev, /lib/udev, /lib/firmware, /usr/share/doc/udev

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
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

| libudev | A library interface to udev device information |
| /etc/udev | Contains Udev configuration files, device permissions, and rules for device naming |

6.62. Vim-7.3

The Vim package contains a powerful text editor.

Approximate build time: 1.1 SBU
Required disk space: 96 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.

6.62.1. 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. A successful test will result in the words "ALL DONE" at completion.

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**.

6.62.2. 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'
```

> **Note**
>
> By default, Vim only installs spell files for the English language. To install spell files for
> your preferred language, download the `*.spl` and optionally, the `*.sug` files for your
> language and character encoding from **ftp://ftp.vim.org/pub/vim/runtime/spell/** and
> save them to `/usr/share/vim/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.

6.62.3. 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

6.63. 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.

6.64. 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 Section 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.

6.65. 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.

Note

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 Section 6.2.2, "Mounting and Populating /dev" and Section 6.2.3, "Mounting Virtual Kernel File Systems".

Chapter 7. Setting Up System Bootscripts

7.1. Introduction

This chapter discusses configuration files and boot scripts. First, the general configuration files needed to set up networking are presented.

- Section 7.2, "General Network Configuration."

- Section 7.3, "Customizing the /etc/hosts File."

Second, issues that affect the proper setup of devices diescussed.

- Section 7.4, "Device and Module Handling on an LFS System."

- Section 7.5, "Creating Custom Symlinks to Devices."

The next sections detail how to install and configure the LFS system scripts needed during the boot process. Most of these scripts will work without modification, but a few require additional configuration files because they deal with hardware-dependent information.

System-V style init scripts are employed in this book because they are widely used and relatively simple. For additional options, a hint detailing the BSD style init setup is available at **http://www.linuxfromscratch.org/hints/downloads/files/bsd-init.txt**. Searching the LFS mailing lists for "depinit", "upstart", or "systemd" will also offer additional information.

If using an alternative style of init scripts, skip these sections.

A listing of the boot scripts are found in Appendix D.

- Section 7.6, "LFS-Bootscripts-20130123."

- Section 7.7, "How Do These Bootscripts Work?."

- Section 7.8, "Configuring the system hostname."

- Section 7.9, "Configuring the setclock Script."

- Section 7.10, "Configuring the Linux Console."

- Section 7.11, "Configuring the sysklogd Script."

Finally, there is a brief introduction to the scripts and configuration files used when the user logs into the system.

- Section 7.13, "The Bash Shell Startup Files."

- Section 7.14, "Creating the /etc/inputrc File."

7.2. General Network Configuration

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, you will need to remove the network symlinks from all run-level directories (/etc/rc.d/rc*.d) after the bootscripts are installed in Section 7.6, "LFS-Bootscripts-20130123".

7.2.1. Creating stable names for network interfaces

If there is only one network interface in the system to be configured, this section is optional, although it will never be wrong to do it. In many cases (e.g. a laptop with a wireless and a wired interface), accomplishing the configuration in this section is necessary.

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.

The rules were pre-generated in the build instructions for udev (systemd) in the last chapter. 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
```

Note

In some cases such as when MAC addresess have been assigned to a network card manually or in a virtual environment such as Xen, the network rules file may not have been generated because addresses are not consistently assigned. In these cases, just continue to the next section.

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.

7.2.2. Creating Network Interface Configuration Files

Which interfaces are brought up and down by the network script depends on the files in /etc/sysconfig/. This directory should contain a file for each interface to be configured, such as ifconfig.xyz, where "xyz" is meaningful to the administrator such as the device name (e.g. eth0). Inside this file are attributes to this interface, such as its IP address(es), subnet masks, and so forth. It is necessary that the stem of the filename be *ifconfig*.

The following command creates a sample file for the *eth0* device with a static IP address:

```
cd /etc/sysconfig/
cat > ifconfig.eth0 << "EOF"
ONBOOT=yes
IFACE=eth0
SERVICE=ipv4-static
IP=192.168.1.1
GATEWAY=192.168.1.2
PREFIX=24
BROADCAST=192.168.1.255
EOF
```

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 automatically brought up. The interface can be manually started or stopped

with the **ifup** and **ifdown** commands.

The IFACE variable defines the interface name, for example, eth0. It is required for all network device configuration files.

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 /lib/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 contains 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.i If omitted, the PREFIX defaults to 24.

For more information see the **ifup** man page.

7.2.3. 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
```

The domain statement can be omitted or replaced with a search statement. See the man page for resolv.conf for more details.

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.

Note

The Google Public IPv4 DNS addresses are 8.8.8.8 and 8.8.4.4.

7.3. Customizing the /etc/hosts File

If a network card is to be configured, decide on the IP address, fully-qualified domain name (FQDN), and possible aliases for use in the /etc/hosts file. The syntax is:

```
IP_address myhost.example.org aliases
```

Unless the computer is to be visible to the Internet (i.e., there is a registered domain and a valid block of assigned IP addresses—most users do not have this), make sure that the IP address is in the private network IP address range. Valid ranges are:

```
Private Network Address Range       Normal Prefix
10.0.0.1 - 10.255.255.254            8
172.x.0.1 - 172.x.255.254           16
192.168.y.1 - 192.168.y.254         24
```

x can be any number in the range 16-31. y can be any number in the range 0-255.

A valid private IP address could be 192.168.1.1. A valid FQDN for this IP could be lfs.example.org.

Even if not using a network card, a valid FQDN is still required. This is necessary for certain programs to operate correctly.

Create the /etc/hosts file by running:

```
cat > /etc/hosts << "EOF"
# Begin /etc/hosts (network card version)

127.0.0.1 localhost
<192.168.1.1> <HOSTNAME.example.org> [alias1] [alias2 ...]

# End /etc/hosts (network card version)
EOF
```

The <192.168.1.1> and <HOSTNAME.example.org> values need to be changed for specific uses or requirements (if assigned an IP address by a network/system administrator and the machine will be connected to an existing network). The optional alias name(s) can be omitted.

If a network card is not going to be configured, create the /etc/hosts file by running:

```
cat > /etc/hosts << "EOF"
# Begin /etc/hosts (no network card version)

127.0.0.1 <HOSTNAME.example.org> <HOSTNAME> localhost

# End /etc/hosts (no network card version)
EOF
```

7.4. Device and Module Handling on an LFS System

In Chapter 6, we installed the Udev package. Before we go into the details regarding how this works, a brief history of previous methods of handling devices is in order.

Linux systems in general traditionally use a static device creation method, whereby a great many device nodes are created under /dev (sometimes literally thousands of nodes), regardless of whether the corresponding hardware devices actually exist. This is typically done via a MAKEDEV script, which

contains a number of calls to the **mknod** program with the relevant major and minor device numbers for every possible device that might exist in the world.

Using the Udev method, only those devices which are detected by the kernel get device nodes created for them. Because these device nodes will be created each time the system boots, they will be stored on a *devtmpfs* file system (a virtual file system that resides entirely in system memory). Device nodes do not require much space, so the memory that is used is negligible.

7.4.1. History

In February 2000, a new filesystem called *devfs* was merged into the 2.3.46 kernel and was made available during the 2.4 series of stable kernels. Although it was present in the kernel source itself, this method of creating devices dynamically never received overwhelming support from the core kernel developers.

The main problem with the approach adopted by *devfs* was the way it handled device detection, creation, and naming. The latter issue, that of device node naming, was perhaps the most critical. It is generally accepted that if device names are allowed to be configurable, then the device naming policy should be up to a system administrator, not imposed on them by any particular developer(s). The *devfs* file system also suffers from race conditions that are inherent in its design and cannot be fixed without a substantial revision to the kernel. It was marked as deprecated for a long period – due to a lack of maintenance – and was finally removed from the kernel in June, 2006.

With the development of the unstable 2.5 kernel tree, later released as the 2.6 series of stable kernels, a new virtual filesystem called *sysfs* came to be. The job of *sysfs* is to export a view of the system's hardware configuration to userspace processes. With this userspace-visible representation, the possibility of seeing a userspace replacement for *devfs* became much more realistic.

7.4.2. Udev Implementation

7.4.2.1. Sysfs

The *sysfs* filesystem was mentioned briefly above. One may wonder how *sysfs* knows about the devices present on a system and what device numbers should be used for them. Drivers that have been compiled into the kernel directly register their objects with a *sysfs* (devtmpfs internally) as they are detected by the kernel. For drivers compiled as modules, this registration will happen when the module is loaded. Once the *sysfs* filesystem is mounted (on /sys), data which the drivers register with *sysfs* are available to userspace processes and to udevd for processing (including modifications to device nodes).

7.4.2.2. Device Node Creation

Device files are created by the kernel by the *devtmpfs* filesystem. Any driver that wishes to register a device node will go through *devtmpfs* (via the driver core) to do it. When a *devtmpfs* instance is mounted on /dev, the device node will initially be created with a fixed name, permissions, and owner.

A short time later, the kernel will send a uevent to **udevd**. Based on the rules specified in the files within the /etc/udev/rules.d, /lib/udev/rules.d, and /run/udev/rules.d directories, **udevd** will create additional symlinks to the device node, or change its permissions, owner, or group, or modify the internal **udevd** database entry (name) for that object.

The rules in these three directories are numbered in a similar fashion to the LFS-Bootscripts package and all three directories are merged together. If **udevd** can't find a rule for the device it is creating, it will leave the permissions and ownership at whatever *devtmpfs* used initially.

7.4.2.3. Udev Bootscripts

The first LFS bootscript, /etc/init.d/mountvirtfs will copy any devices located in /lib/udev /devices to /dev. This is necessary because some devices, directories, and symlinks are needed before the dynamic device handling processes are available during the early stages of booting a system, or are required by **udevd** itself. Creating static device nodes in /lib/udev/devices also provides an easy workaround for devices that are not supported by the dynamic device handling infrastructure.

The /etc/rc.d/init.d/udev initscript starts **udevd**, triggers any "coldplug" devices that have already been created by the kernel and waits for any rules to complete. The script also unsets the uevent handler from the default of /sbin/hotplug . This is done because the kernel no longer needs to call out to an external binary. Instead **udevd** will listen on a netlink socket for uevents that the kernel raises.

The **/etc/rc.d/init.d/udev_retry** initscript takes care of re-triggering events for subsystems whose rules may rely on filesystems that are not mounted until the **mountfs** script is run (in particular, /usr and /var may cause this). This script runs after the **mountfs** script, so those rules (if re-triggered) should succeed the second time around. It is configured from the /etc/sysconfig /udev_retry file; any words in this file other than comments are considered subsystem names to trigger at retry time. To find the subsystem of a device, use **udevadm info --attribute-walk <device>** where <device> is a an absolure path in /dev or /sys such as /dev/sr0 or /sys/class/rtc.

7.4.2.4. Module Loading

Device drivers compiled as modules may have aliases built into them. Aliases are visible in the output of the **modinfo** program and are usually related to the bus-specific identifiers of devices supported by a module. For example, the *snd-fm801* driver supports PCI devices with vendor ID 0x1319 and device ID 0x0801, and has an alias of "pci:v00001319d00000801sv*sd*bc04sc01i*". For most devices, the bus driver exports the alias of the driver that would handle the device via *sysfs*. E.g., the /sys/bus/pci/devices/0000:00:0d.0/modalias file might contain the string "pci:v00001319d00000801sv00001319sd00001319bc04sc01i00". The default rules provided with Udev will cause **udevd** to call out to **/sbin/modprobe** with the contents of the MODALIAS uevent environment variable (which should be the same as the contents of the modalias file in sysfs), thus loading all modules whose aliases match this string after wildcard expansion.

In this example, this means that, in addition to *snd-fm801*, the obsolete (and unwanted) *forte* driver will be loaded if it is available. See below for ways in which the loading of unwanted drivers can be prevented.

The kernel itself is also able to load modules for network protocols, filesystems and NLS support on demand.

7.4.2.5. Handling Hotpluggable/Dynamic Devices

When you plug in a device, such as a Universal Serial Bus (USB) MP3 player, the kernel recognizes that the device is now connected and generates a uevent. This uevent is then handled by **udevd** as described above.

7.4.3. Problems with Loading Modules and Creating Devices

There are a few possible problems when it comes to automatically creating device nodes.

7.4.3.1. A kernel module is not loaded automatically

Udev will only load a module if it has a bus-specific alias and the bus driver properly exports the necessary aliases to *sysfs*. In other cases, one should arrange module loading by other means. With Linux-3.8.1, Udev is known to load properly-written drivers for INPUT, IDE, PCI, USB, SCSI,

SERIO, and FireWire devices.

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-3.8.1, this is the case with ISA busses. Expect this issue to be fixed in later kernel versions.

Udev is not intended to load "wrapper" drivers such as *snd-pcm-oss* and non-hardware drivers such as *loop* at all.

7.4.3.2. A kernel module is not loaded automatically, and Udev is not intended to load it

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 a "softdep" line in any /etc/modprobe.d/*<filename>*.conf file. For example:

```
softdep snd-pcm post: snd-pcm-oss
```

Note that the "softdep" command also allows pre: dependencies, or a mixture of both pre: and post:. See the modprobe.d(5) manual page for more information on "softdep" syntax and capabilities.

If the module in question is not a wrapper and is useful by itself, configure the **modules** 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.

7.4.3.3. Udev loads some unwanted module

Either don't build the module, or blacklist it in a /etc/modprobe.d/blacklist.conf file as done with the *forte* module in the example below:

```
blacklist forte
```

Blacklisted modules can still be loaded manually with the explicit **modprobe** command.

7.4.3.4. Udev creates a device incorrectly, or makes a wrong symlink

This usually happens if a rule unexpectedly matches a device. For example, a poorly-writen rule can match both a SCSI disk (as desired) and the corresponding SCSI generic device (incorrectly) by vendor. Find the offending rule and make it more specific, with the help of the **udevadm info** command.

7.4.3.5. Udev rule works unreliably

This may be another manifestation of the previous problem. If not, and your rule uses *sysfs* attributes, it may be a kernel timing issue, to be fixed in later kernels. For now, you can work around it by creating a rule that waits for the used *sysfs* attribute and appending it to the /etc/udev

/rules.d/10-wait_for_sysfs.rules file (create this file if it does not exist). Please notify the LFS Development list if you do so and it helps.

7.4.3.6. Udev does not create a device

Further text assumes that the driver is built statically into the kernel or already loaded as a module, and that you have already checked that Udev doesn't create a misnamed device.

Udev has no information needed to create a device node if a kernel driver does not export its data to *sysfs*. This is most common with third party drivers from outside the kernel tree. Create a static device node in /lib/udev/devices with the appropriate major/minor numbers (see the file devices.txt inside the kernel documentation or the documentation provided by the third party driver vendor). The static device node will be copied to /dev by the **udev** bootscript.

7.4.3.7. Device naming order changes randomly after rebooting

This is due to the fact that Udev, by design, handles uevents and loads modules in parallel, and thus in an unpredictable order. This will never be "fixed". You should not rely upon the kernel device names being stable. Instead, create your own rules that make symlinks with stable names based on some stable attributes of the device, such as a serial number or the output of various *_id utilities installed by Udev. See Section 7.5, "Creating Custom Symlinks to Devices" and Section 7.2, "General Network Configuration" for examples.

7.4.4. Useful Reading

Additional helpful documentation is available at the following sites:

* A Userspace Implementation of *devfs* **http://www.kroah.com/linux/talks /ols_2003_udev_paper/Reprint-Kroah-Hartman-OLS2003.pdf**

* The *sysfs* Filesystem **http://www.kernel.org/pub/linux/kernel/people/mochel/doc/papers /ols-2005/mochel.pdf**

7.5. Creating Custom Symlinks to Devices

7.5.1. CD-ROM symlinks

Some software that you may want to install later (e.g., various media players) expect the /dev/cdrom and /dev/dvd symlinks to exist, and to point to a CD-ROM or DVD-ROM device. Also, it may be convenient to put references to those symlinks into /etc/fstab. Udev comes with a script that will generate rules files to create these symlinks for you, depending on the capabilities of each device, but you need to decide which of two modes of operation you wish to have the script use.

First, the script can operate in "by-path" mode (used by default for USB and FireWire devices), where the rules it creates depend on the physical path to the CD or DVD device. Second, it can operate in "by-id" mode (default for IDE and SCSI devices), where the rules it creates depend on identification strings stored in the CD or DVD device itself. The path is determined by Udev's **path_id** script, and the identification strings are read from the hardware by its **ata_id** or **scsi_id** programs, depending on which type of device you have.

There are advantages to each approach; the correct approach to use will depend on what kinds of device changes may happen. If you expect the physical path to the device (that is, the ports and/or slots that it plugs into) to change, for example because you plan on moving the drive to a different IDE port or a different USB connector, then you should use the "by-id" mode. On the other hand, if you expect the device's identification to change, for example because it may die, and you would

replace it with a different device with the same capabilities and which is plugged into the same connectors, then you should use the "by-path" mode.

If either type of change is possible with your drive, then choose a mode based on the type of change you expect to happen more often.

> **Important**
>
> 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 /etc/udev/rules.d/83-cdrom-symlinks.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"/' \
    /etc/udev/rules.d/83-cdrom-symlinks.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.

7.5.2. Dealing with duplicate devices

As explained in Section 7.4, "Device and Module Handling on an LFS System", 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 Section 7.2, "General Network Configuration", 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/video`*x*. 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.

7.6. LFS-Bootscripts-20130123

The LFS-Bootscripts package contains a set of scripts to start/stop the LFS system at bootup/shutdown.

Approximate build time:	less than 0.1 SBU
Required disk space:	256 KB

7.6.1. Installation of LFS-Bootscripts

Install the package:

```
make install
```

7.6.2. Contents of LFS-Bootscripts

Installed scripts:	checkfs, cleanfs, console, functions, halt, ifdown, ifup, localnet, modules, mountfs, mountkernfs, network, rc, reboot, sendsignals, setclock, static, swap, sysctl, sysklogd, template, udev, and udev_retry
Installed directories:	/etc/rc.d, /etc/init.d (symbolic link), /etc/sysconfig, /lib/services, /lib/lsb (symbolic link)

Short Descriptions

checkfs	Checks the integrity of the file systems before they are mounted (with the exception of journal and network based file systems)
cleanfs	Removes files that should not be preserved between reboots, such as those in /var/run/ and /var/lock/; it re-creates /var/run/utmp and removes the possibly present /etc/nologin, /fastboot, and /forcefsck files

console	Loads the correct keymap table for the desired keyboard layout; it also sets the screen font
functions	Contains common functions, such as error and status checking, that are used by several bootscripts
halt	Halts the system
ifdown	Stops a network device
ifup	Initializes a network device
localnet	Sets up the system's hostname and local loopback device
modules	Loads kernel modules listed in /etc/sysconfig/modules, using arguments that are also given there
mountfs	Mounts all file systems, except ones that are marked *noauto* or are network based
mountkernfs	Mounts virtual kernel file systems, such as *proc*
network	Sets up network interfaces, such as network cards, and sets up the default gateway (where applicable)
rc	The master run-level control script; it is responsible for running all the other bootscripts one-by-one, in a sequence determined by the name of the symbolic links being processed
reboot	Reboots the system
sendsignals	Makes sure every process is terminated before the system reboots or halts
setclock	Resets the kernel clock to local time in case the hardware clock is not set to UTC time
static	Provides the functionality needed to assign a static Internet Protocol (IP) address to a network interface
swap	Enables and disables swap files and partitions
sysctl	Loads system configuration values from /etc/sysctl.conf, if that file exists, into the running kernel
sysklogd	Starts and stops the system and kernel log daemons
template	A template to create custom bootscripts for other daemons
udev	Prepares the /dev directory and starts Udev
udev_retry	Retries failed udev uevents, and copies generated rules files from to /etc/udev/rules.d if required

7.7. How Do These Bootscripts Work?

Linux uses a special booting facility named SysVinit that is based on a concept of *run-levels*. It can be quite different from one system to another, so it cannot be assumed that because things worked in one particular Linux distribution, they should work the same in LFS too. LFS has its own way of doing things, but it respects generally accepted standards.

SysVinit (which will be referred to as "init" from now on) works using a run-levels scheme. There are seven (numbered 0 to 6) run-levels (actually, there are more run-levels, but they are for special cases and are generally not used. See init(8) for more details), and each one of those corresponds to the actions the computer is supposed to perform when it starts up. The default run-level is 3. Here are the

descriptions of the different run-levels as they are implemented:

0: halt the computer
1: single-user mode
2: multi-user mode without networking
3: multi-user mode with networking
4: reserved for customization, otherwise does the same as 3
5: same as 4, it is usually used for GUI login (like X's **xdm** or KDE's **kdm**)
6: reboot the computer

7.7.1. Configuring Sysvinit

During the kernel initialization, the first program that is run is either specified on the command line or, by default **init**. This program reads the initialization file /etc/inittab. Create this file with:

```
cat > /etc/inittab << "EOF"
# Begin /etc/inittab

id:3:initdefault:

si::sysinit:/etc/rc.d/init.d/rc S

l0:0:wait:/etc/rc.d/init.d/rc 0
l1:S1:wait:/etc/rc.d/init.d/rc 1
l2:2:wait:/etc/rc.d/init.d/rc 2
l3:3:wait:/etc/rc.d/init.d/rc 3
l4:4:wait:/etc/rc.d/init.d/rc 4
l5:5:wait:/etc/rc.d/init.d/rc 5
l6:6:wait:/etc/rc.d/init.d/rc 6

ca:12345:ctrlaltdel:/sbin/shutdown -t1 -a -r now

su:S016:once:/sbin/sulogin

1:2345:respawn:/sbin/agetty --noclear tty1 9600
2:2345:respawn:/sbin/agetty tty2 9600
3:2345:respawn:/sbin/agetty tty3 9600
4:2345:respawn:/sbin/agetty tty4 9600
5:2345:respawn:/sbin/agetty tty5 9600
6:2345:respawn:/sbin/agetty tty6 9600

# End /etc/inittab
EOF
```

An explanation of this initialization file is in the man page for *inittab*. For LFS, the key command that is run is **rc**. The intialization file above will instruct **rc** to run all the scripts starting with an S in the /etc/rc.d/rcsysinit.d directory followed by all the scripts starting with an S in the /etc/rc.d/rc?.d directory where the question mark is specified by the initdefault value.

As a convenience, the **rc** script reads a library of functions in /lib/lsb/init-functions. This library also reads an optional configuration file, /etc/sysconfig/rc.site. Any of the system configuration file parameters described in subsequent sections can be alternatively placed in this file allowing consolidation of all system parameters in this one file.

As a debugging convenience, the functions script also logs all output to /run/var/bootlog. Since

the /run directory is a tmpfs, this file is not persistent across boots, however it is appended to the more permanent file /var/log/boot.log at the end of the boot process.

7.7.2. Changing Run Levels

Changing run-levels is done with **init <runlevel>**, where *<runlevel>* is the target run-level. For example, to reboot the computer, a user could issue the **init 6** command, which is an alias for the **reboot** command. Likewise, **init 0** is an alias for the **halt** command.

There are a number of directories under /etc/rc.d that look like rc?.d (where ? is the number of the run-level) and rcsysinit.d, all containing a number of symbolic links. Some begin with a *K*, the others begin with an *S*, and all of them have two numbers following the initial letter. The K means to stop (kill) a service and the S means to start a service. The numbers determine the order in which the scripts are run, from 00 to 99—the lower the number the earlier it gets executed. When **init** switches to another run-level, the appropriate services are either started or stopped, depending on the runlevel chosen.

The real scripts are in /etc/rc.d/init.d. They do the actual work, and the symlinks all point to them. K links and S links point to the same script in /etc/rc.d/init.d. This is because the scripts can be called with different parameters like *start*, *stop*, *restart*, *reload*, and *status*. When a K link is encountered, the appropriate script is run with the *stop* argument. When an S link is encountered, the appropriate script is run with the *start* argument.

There is one exception to this explanation. Links that start with an *S* in the rc0.d and rc6.d directories will not cause anything to be started. They will be called with the parameter *stop* to stop something. The logic behind this is that when a user is going to reboot or halt the system, nothing needs to be started. The system only needs to be stopped.

These are descriptions of what the arguments make the scripts do:

start

The service is started.

stop

The service is stopped.

restart

The service is stopped and then started again.

reload

The configuration of the service is updated. This is used after the configuration file of a service was modified, when the service does not need to be restarted.

status

Tells if the service is running and with which PIDs.

Feel free to modify the way the boot process works (after all, it is your own LFS system). The files given here are an example of how it can be done.

7.8. Configuring the system hostname

Part of the job of the **localnet** script is setting the system's hostname. This needs to be configured in the /etc/sysconfig/network file.

Create the /etc/sysconfig/network file and enter a hostname by running:

```
echo "HOSTNAME=<lfs>" > /etc/sysconfig/network
```

<lfs> needs to be replaced with the name given to the computer. Do not enter the Fully Qualified Domain Name (FQDN) here. That information is put in the /etc/hosts file.

7.9. Configuring the setclock Script

The **setclock** script reads the time from the hardware clock, also known as the BIOS or the Complementary Metal Oxide Semiconductor (CMOS) clock. If the hardware clock is set to UTC, this script will convert the hardware clock's time to the local time using the /etc/localtime file (which tells the **hwclock** program which timezone the user is in). There is no way to detect whether or not the hardware clock is set to UTC, so this needs to be configured manually.

The **setclock** is run via udev when the kernel detects the hardware capability upon boot. It can also be run manually with the stop parameter to store the system time to the CMOS clock.

If you cannot remember whether or not the hardware clock is set to UTC, find out by running the **hwclock --localtime --show** command. This will display what the current time is according to the hardware clock. If this time matches whatever your watch says, then the hardware clock is set to local time. If the output from **hwclock** is not local time, chances are it is set to UTC time. Verify this by adding or subtracting the proper amount of hours for the timezone to the time shown by **hwclock**. For example, if you are currently in the MST timezone, which is also known as GMT -0700, add seven hours to the local time.

Change the value of the UTC variable below to a value of *0* (zero) if the hardware clock is *not* set to UTC time.

Create a new file /etc/sysconfig/clock by running the following:

```
cat > /etc/sysconfig/clock << "EOF"
# Begin /etc/sysconfig/clock

UTC=1

# Set this to any options you might need to give to hwclock,
# such as machine hardware clock type for Alphas.
CLOCKPARAMS=

# End /etc/sysconfig/clock
EOF
```

A good hint explaining how to deal with time on LFS is available at **http://www.linuxfromscratch.org/hints/downloads/files/time.txt**. It explains issues such as time zones, UTC, and the TZ environment variable.

Note

The CLOCKPARAMS and UTC paramaters may be alternatively set in the /etc/sysconfig/rc.site file.

7.10. Configuring the Linux Console

This section discusses how to configure the **console** bootscript that sets up the keyboard map, console font and console kernel log level. If non-ASCII characters (e.g., the copyright sign, the British pound sign and Euro symbol) will not be used and the keyboard is a U.S. one, much of this section can be skipped. Without the configuration file, (or equivalent settings in rc.site), the **console** bootscript will do nothing.

The **console** script reads the /etc/sysconfig/console file for configuration information. Decide which keymap and screen font will be used. Various language-specific HOWTOs can also help with this, see **http://www.tldp.org/HOWTO/HOWTO-INDEX/other-lang.html**. If still in doubt, look in the /lib/kbd directory for valid keymaps and screen fonts. Read loadkeys(1) and setfont(8) manual pages to determine the correct arguments for these programs.

The /etc/sysconfig/console file should contain lines of the form: VARIABLE="value". The following variables are recognized:

LOGLEVEL
This variable specifies the log level for kernel messages sent to the console as set by **dmesg**. Valid levels are from "1" (no messages) to "8". The default level is "7".

KEYMAP
This variable specifies the arguments for the **loadkeys** program, typically, the name of keymap to load, e.g., "es". If this variable is not set, the bootscript will not run the **loadkeys** program, and the default kernel keymap will be used.

KEYMAP_CORRECTIONS
This (rarely used) variable specifies the arguments for the second call to the **loadkeys** program. This is useful if the stock keymap is not completely satisfactory and a small adjustment has to be made. E.g., to include the Euro sign into a keymap that normally doesn't have it, set this variable to "euro2".

FONT
This variable specifies the arguments for the **setfont** program. Typically, this includes the font name, "-m", and the name of the application character map to load. E.g., in order to load the "lat1-16" font together with the "8859-1" application character map (as it is appropriate in the USA), set this variable to "lat1-16 -m 8859-1". In UTF-8 mode, the kernel uses the application character map for conversion of composed 8-bit key codes in the keymap to UTF-8, and thus the argument of the "-m" parameter should be set to the encoding of the composed key codes in the keymap.

UNICODE
Set this variable to "1", "yes" or "true" in order to put the console into UTF-8 mode. This is useful in UTF-8 based locales and harmful otherwise.

LEGACY_CHARSET
For many keyboard layouts, there is no stock Unicode keymap in the Kbd package. The **console** bootscript will convert an available keymap to UTF-8 on the fly if this variable is set to the encoding of the available non-UTF-8 keymap.

Some examples:

- For a non-Unicode setup, only the KEYMAP and FONT variables are generally needed. E.g., for a Polish setup, one would use:

```
cat > /etc/sysconfig/console << "EOF"
# Begin /etc/sysconfig/console
```

```
KEYMAP="pl2"
FONT="lat2a-16 -m 8859-2"

# End /etc/sysconfig/console
EOF
```

- As mentioned above, it is sometimes necessary to adjust a stock keymap slightly. The following example adds the Euro symbol to the German keymap:

```
cat > /etc/sysconfig/console << "EOF"
# Begin /etc/sysconfig/console

KEYMAP="de-latin1"
KEYMAP_CORRECTIONS="euro2"
FONT="lat0-16 -m 8859-15"

# End /etc/sysconfig/console
EOF
```

- The following is a Unicode-enabled example for Bulgarian, where a stock UTF-8 keymap exists:

```
cat > /etc/sysconfig/console << "EOF"
# Begin /etc/sysconfig/console

UNICODE="1"
KEYMAP="bg_bds-utf8"
FONT="LatArCyrHeb-16"

# End /etc/sysconfig/console
EOF
```

- Due to the use of a 512-glyph LatArCyrHeb-16 font in the previous example, bright colors are no longer available on the Linux console unless a framebuffer is used. If one wants to have bright colors without framebuffer and can live without characters not belonging to his language, it is still possible to use a language-specific 256-glyph font, as illustrated below:

```
cat > /etc/sysconfig/console << "EOF"
# Begin /etc/sysconfig/console

UNICODE="1"
KEYMAP="bg_bds-utf8"
FONT="cyr-sun16"

# End /etc/sysconfig/console
EOF
```

- The following example illustrates keymap autoconversion from ISO-8859-15 to UTF-8 and enabling dead keys in Unicode mode:

```
cat > /etc/sysconfig/console << "EOF"
# Begin /etc/sysconfig/console

UNICODE="1"
KEYMAP="de-latin1"
```

```
KEYMAP_CORRECTIONS="euro2"
LEGACY_CHARSET="iso-8859-15"
FONT="LatArCyrHeb-16 -m 8859-15"

# End /etc/sysconfig/console
EOF
```

- Some keymaps have dead keys (i.e., keys that don't produce a character by themselves, but put an accent on the character produced by the next key) or define composition rules (such as: "press Ctrl+. A E to get Æ" in the default keymap). Linux-3.8.1 interprets dead keys and composition rules in the keymap correctly only when the source characters to be composed together are not multibyte. This deficiency doesn't affect keymaps for European languages, because there accents are added to unaccented ASCII characters, or two ASCII characters are composed together. However, in UTF-8 mode it is a problem, e.g., for the Greek language, where one sometimes needs to put an accent on the letter "alpha". The solution is either to avoid the use of UTF-8, or to install the X window system that doesn't have this limitation in its input handling.

- For Chinese, Japanese, Korean and some other languages, the Linux console cannot be configured to display the needed characters. Users who need such languages should install the X Window System, fonts that cover the necessary character ranges, and the proper input method (e.g., SCIM, it supports a wide variety of languages).

> **Note**
>
> The /etc/sysconfig/console file only controls the Linux text console localization. It has nothing to do with setting the proper keyboard layout and terminal fonts in the X Window System, with ssh sessions or with a serial console. In such situations, limitations mentioned in the last two list items above do not apply.

7.11. Configuring the sysklogd Script

The sysklogd script invokes the **syslogd** program with the *-m 0* option. This option turns off the periodic timestamp mark that **syslogd** writes to the log files every 20 minutes by default. If you want to turn on this periodic timestamp mark, edit /etc/sysconfig/rc.site and define the variable SYSKLOGD_PARMS to the desired value. For instance, to remove all parameters, set the variable to a null value:

```
SYSKLOGD_PARMS=
```

See **man syslogd** for more options.

7.12. The rc.site File

The optional /etc/sysconfig/rc.site file contains settings that are automatically set for each boot script. It can alternatively set the values specified in the hostname, console, and clock files in the /etc/sysconfig/ directory. If the associated variables are present in both these separate files and rc.site, the values in the script specific files have precedence.

rc.site also contains parameters that can customize other aspects of the boot process. Setting the IPROMPT variable will enable selective running of bootscripts. Other options are described in the file comments. The default version of the file is as follows:

```
# rc.site
# Optional parameters for boot scripts.

# Distro Information
# These values, if specified here, override the defaults
#DISTRO="Linux From Scratch" # The distro name
#DISTRO_CONTACT="lfs-dev@linuxfromscratch.org" # Bug report address
#DISTRO_MINI="LFS" # Short name used in filenames for distro config

# Define custom colors used in messages printed to the screen

# Please consult `man console_codes` for more information
# under the "ECMA-48 Set Graphics Rendition" section
#
# Warning: when switching from a 8bit to a 9bit font,
# the linux console will reinterpret the bold (1;) to
# the top 256 glyphs of the 9bit font.  This does
# not affect framebuffer consoles

# These values, if specified here, override the defaults
#BRACKET="\\033[1;34m" # Blue
#FAILURE="\\033[1;31m" # Red
#INFO="\\033[1;36m"    # Cyan
#NORMAL="\\033[0;39m"  # Grey
#SUCCESS="\\033[1;32m" # Green
#WARNING="\\033[1;33m" # Yellow

# Use a colored prefix
# These values, if specified here, override the defaults
#BMPREFIX="      "
#SUCCESS_PREFIX="${SUCCESS}  *  ${NORMAL}"
#FAILURE_PREFIX="${FAILURE}*****${NORMAL}"
#WARNING_PREFIX="${WARNING} *** ${NORMAL}"

# Interactive startup
#IPROMPT="yes" # Whether to display the interactive boot prompt
#itime="3"    # The amount of time (in seconds) to display the prompt

# The total length of the distro welcome string, without escape codes
#wlen=$(echo "Welcome to ${DISTRO}" | wc -c )
#welcome_message="Welcome to ${INFO}${DISTRO}${NORMAL}"

# The total length of the interactive string, without escape codes
#ilen=$(echo "Press 'I' to enter interactive startup" | wc -c )
#i_message="Press '${FAILURE}I${NORMAL}' to enter interactive startup"

# Set scripts to skip the file system check on reboot
#FASTBOOT=yes

# Skip reading from the console
#HEADLESS=yes

# Speed up boot without waiting for settle in udev_retry
```

```
#OMIT_UDEV_RETRY_SETTLE=yes

# Write out fsck progress if yes
#VERBOSE_FSCK=no

# Speed up boot without waiting for settle in udev
#OMIT_UDEV_SETTLE=y

# Speed up boot without waiting for settle in udev_retry
#OMIT_UDEV_RETRY_SETTLE=yes

# Skip cleaning /tmp if yes
#SKIPTMPCLEAN=no

# For setclock
#UTC=1
#CLOCKPARAMS=

# For consolelog
#LOGLEVEL=5

# For network
#HOSTNAME=mylfs

# Delay between TERM and KILL signals at shutdown
#KILLDELAY=3

# Optional sysklogd parameters
#SYSKLOGD_PARMS="-m 0"

# Console parameters
#UNICODE=1
#KEYMAP="de-latin1"
#KEYMAP_CORRECTIONS="euro2"
#FONT="lat0-16 -m 8859-15"
#LEGACY_CHARSET=
```

7.12.1. Customizing the Boot and Shutdown Scripts

The LFS boot scripts boot and shut down a system in a fairly efficient manner, but there are a few tweaks that you can make in the rc.site file to improve speed even more and to adjust messages according to your preferences. To do this, adjust the settings in the /etc/sysconfig/rc.site file above.

- During the boot script udev, there is a call to **udev settle** that requires some time to complete. This time may or may not be required depending on devices present in the system. If you only have simple partitions and a single ethernet card, the boot process will probably not need to wait for this command. To skip it, set the variable OMIT_UDEV_SETTLE=y.

- The boot script udev_retry also runs **udev settle** by default. This command is only needed by default if the /var directory is separately mounted. This is because the clock needs the file /var/lib/hwclock/adjtime. Other customizations may also need to wait for udev to complete, but in many installations it is not needed. Skip the command by setting the variable OMIT_UDEV_RETRY_SETTLE=y.

- By default, the file system checks are silent. This can appear to be a delay during the bootup

process. To turn on the **fsck** output, set the variable VERBOSE_FSCK=y.

- When rebooting, you may want to skip the filesystem check, **fsck**, completely. To do this, either create the file /fastboot or reboot the system with the command **/sbin/shutdown -f -r now**. On the other hand, you can force all file systems to be checked by creating /forcefsck or running **shutdown** with the *-F* parameter instead of *-f*.

  Setting the variable FASTBOOT=y will disable **fsck** during the boot process until it is removed. This is not recommended on a permanent basis.

- Normally, all files in the /tmp directory are deleted at boot time. Depending on the number of files or directories present, this can cause a noticeable delay in the boot process. To skip removing these files set the variable SKIPTMPCLEAN=y.

- During shutdown, the **init** program sends a TERM signal to each program it has started (e.g. agetty), waits for a set time (default 3 seconds), and sends each process a KILL signal and waits again. This process is repeated in the **sendsignals** script for any processes that are not shut down by their own scripts. The delay for **init** can be set by passing a parameter. For example to remove the delay in **init**, pass the -t0 parameter when shutting down or rebooting (e.g. **/sbin/shutdown -t0 -r now**). The delay for the **sendsignals** script can be skipped by setting the parameter KILLDELAY=0.

7.13. The Bash Shell Startup Files

The shell program **/bin/bash** (hereafter referred to as "the shell") uses a collection of startup files to help create an environment to run in. Each file has a specific use and may affect login and interactive environments differently. The files in the /etc directory provide global settings. If an equivalent file exists in the home directory, it may override the global settings.

An interactive login shell is started after a successful login, using **/bin/login**, by reading the /etc/passwd file. An interactive non-login shell is started at the command-line (e.g., [prompt]$**/bin /bash**). A non-interactive shell is usually present when a shell script is running. It is non-interactive because it is processing a script and not waiting for user input between commands.

For more information, see **info bash** under the *Bash Startup Files and Interactive Shells* section.

The files /etc/profile and ~/.bash_profile are read when the shell is invoked as an interactive login shell.

The base /etc/profile below sets some environment variables necessary for native language support. Setting them properly results in:

- The output of programs translated into the native language

- Correct classification of characters into letters, digits and other classes. This is necessary for **bash** to properly accept non-ASCII characters in command lines in non-English locales

- The correct alphabetical sorting order for the country

- Appropriate default paper size

- Correct formatting of monetary, time, and date values

Replace *<ll>* below with the two-letter code for the desired language (e.g., "en") and *<CC>* with the two-letter code for the appropriate country (e.g., "GB"). *<charmap>* should be replaced with the canonical charmap for your chosen locale. Optional modifiers such as "@euro" may also be present.

The list of all locales supported by Glibc can be obtained by running the following command:

```
locale -a
```

Charmaps can have a number of aliases, e.g., "ISO-8859-1" is also referred to as "iso8859-1" and "iso88591". Some applications cannot handle the various synonyms correctly (e.g., require that "UTF-8" is written as "UTF-8", not "utf8"), so it is safest in most cases to choose the canonical name for a particular locale. To determine the canonical name, run the following command, where *<locale name>* is the output given by **locale -a** for your preferred locale ("en_GB.iso88591" in our example).

```
LC_ALL=<locale name> locale charmap
```

For the "en_GB.iso88591" locale, the above command will print:

```
ISO-8859-1
```

This results in a final locale setting of "en_GB.ISO-8859-1". It is important that the locale found using the heuristic above is tested prior to it being added to the Bash startup files:

```
LC_ALL=<locale name> locale language
LC_ALL=<locale name> locale charmap
LC_ALL=<locale name> locale int_curr_symbol
LC_ALL=<locale name> locale int_prefix
```

The above commands should print the language name, the character encoding used by the locale, the local currency, and the prefix to dial before the telephone number in order to get into the country. If any of the commands above fail with a message similar to the one shown below, this means that your locale was either not installed in Chapter 6 or is not supported by the default installation of Glibc.

```
locale: Cannot set LC_* to default locale: No such file or directory
```

If this happens, you should either install the desired locale using the **localedef** command, or consider choosing a different locale. Further instructions assume that there are no such error messages from Glibc.

Some packages beyond LFS may also lack support for your chosen locale. One example is the X library (part of the X Window System), which outputs the following error message if the locale does not exactly match one of the character map names in its internal files:

```
Warning: locale not supported by Xlib, locale set to C
```

In several cases Xlib expects that the character map will be listed in uppercase notation with canonical dashes. For instance, "ISO-8859-1" rather than "iso88591". It is also possible to find an appropriate specification by removing the charmap part of the locale specification. This can be checked by running the **locale charmap** command in both locales. For example, one would have to change "de_DE.ISO-8859-15@euro" to "de_DE@euro" in order to get this locale recognized by Xlib.

Other packages can also function incorrectly (but may not necessarily display any error messages) if the locale name does not meet their expectations. In those cases, investigating how other Linux distributions support your locale might provide some useful information.

Once the proper locale settings have been determined, create the /etc/profile file:

```
cat > /etc/profile << "EOF"
```

```
# Begin /etc/profile


export LANG=<ll>_<CC>.<charmap><@modifiers>


# End /etc/profile
EOF
```

The "C" (default) and "en_US" (the recommended one for United States English users) locales are different. "C" uses the US-ASCII 7-bit character set, and treats bytes with the high bit set as invalid characters. That's why, e.g., the `ls` command substitutes them with question marks in that locale. Also, an attempt to send mail with such characters from Mutt or Pine results in non-RFC-conforming messages being sent (the charset in the outgoing mail is indicated as "unknown 8-bit"). So you can use the "C" locale only if you are sure that you will never need 8-bit characters.

UTF-8 based locales are not supported well by many programs. Work is in progress to document and, if possible, fix such problems, see **http://www.linuxfromscratch.org/blfs/view/svn/introduction /locale-issues.html**.

7.14. Creating the /etc/inputrc File

The `inputrc` file handles keyboard mapping for specific situations. This file is the startup file used by Readline — the input-related library — used by Bash and most other shells.

Most people do not need user-specific keyboard mappings so the command below creates a global /etc/inputrc used by everyone who logs in. If you later decide you need to override the defaults on a per-user basis, you can create a .inputrc file in the user's home directory with the modified mappings.

For more information on how to edit the inputrc file, see **info bash** under the *Readline Init File* section. **info readline** is also a good source of information.

Below is a generic global inputrc along with comments to explain what the various options do. Note that comments cannot be on the same line as commands. Create the file using the following command:

```
cat > /etc/inputrc << "EOF"
# Begin /etc/inputrc
# Modified by Chris Lynn <roryo@roryo.dynup.net>

# Allow the command prompt to wrap to the next line
set horizontal-scroll-mode Off

# Enable 8bit input
set meta-flag On
set input-meta On

# Turns off 8th bit stripping
set convert-meta Off

# Keep the 8th bit for display
set output-meta On

# none, visible or audible
set bell-style none
```

```
# All of the following map the escape sequence of the value
# contained in the 1st argument to the readline specific functions
"\eOd": backward-word
"\eOc": forward-word

# for linux console
"\e[1~": beginning-of-line
"\e[4~": end-of-line
"\e[5~": beginning-of-history
"\e[6~": end-of-history
"\e[3~": delete-char
"\e[2~": quoted-insert

# for xterm
"\eOH": beginning-of-line
"\eOF": end-of-line

# for Konsole
"\e[H": beginning-of-line
"\e[F": end-of-line

# End /etc/inputrc
EOF
```

Chapter 8. Making the LFS System Bootable

8.1. Introduction

It is time to make the LFS system bootable. This chapter discusses creating an fstab file, building a kernel for the new LFS system, and installing the GRUB boot loader so that the LFS system can be selected for booting at startup.

8.2. Creating the /etc/fstab File

The /etc/fstab file is used by some programs to determine where file systems are to be mounted by default, in which order, and which must be checked (for integrity errors) prior to mounting. Create a new file systems table like this:

```
cat > /etc/fstab << "EOF"
# Begin /etc/fstab

# file system  mount-point  type     options             dump  fsck
#                                                                order

/dev/<xxx>     /            <fff>    defaults            1     1
/dev/<yyy>     swap         swap     pri=1               0     0
proc           /proc        proc     nosuid,noexec,nodev 0     0
sysfs          /sys         sysfs    nosuid,noexec,nodev 0     0
devpts         /dev/pts     devpts   gid=5,mode=620      0     0
```

```
tmpfs           /run         tmpfs    defaults          0    0
devtmpfs        /dev         devtmpfs mode=0755,nosuid   0    0

# End /etc/fstab
EOF
```

Replace *<xxx>*, *<yyy>*, and *<fff>* with the values appropriate for the system, for example, hda2, hda5, and *ext3*. For details on the six fields in this file, see **man 5 fstab**.

Filesystems with MS-DOS or Windows origin (i.e.: vfat, ntfs, smbfs, cifs, iso9660, udf) need the "iocharset" mount option in order for non-ASCII characters in file names to be interpreted properly. The value of this option should be the same as the character set of your locale, adjusted in such a way that the kernel understands it. This works if the relevant character set definition (found under File systems -> Native Language Support) has been compiled into the kernel or built as a module. The "codepage" option is also needed for vfat and smbfs filesystems. It should be set to the codepage number used under MS-DOS in your country. E.g., in order to mount USB flash drives, a ru_RU.KOI8-R user would need the following in the options portion of its mount line in /etc/fstab:

```
noauto,user,quiet,showexec,iocharset=koi8r,codepage=866
```

The corresponding options fragment for ru_RU.UTF-8 users is:

```
noauto,user,quiet,showexec,iocharset=utf8,codepage=866
```

> **Note**
>
> In the latter case, the kernel emits the following message:
>
> ```
> FAT: utf8 is not a recommended IO charset for FAT filesystems,
>     filesystem will be case sensitive!
> ```
>
> This negative recommendation should be ignored, since all other values of the "iocharset" option result in wrong display of filenames in UTF-8 locales.

It is also possible to specify default codepage and iocharset values for some filesystems during kernel configuration. The relevant parameters are named "Default NLS Option" (CONFIG_NLS_DEFAULT), "Default Remote NLS Option" (CONFIG_SMB_NLS_DEFAULT), "Default codepage for FAT" (CONFIG_FAT_DEFAULT_CODEPAGE), and "Default iocharset for FAT" (CONFIG_FAT_DEFAULT_IOCHARSET). There is no way to specify these settings for the ntfs filesystem at kernel compilation time.

It is possible to make the ext3 filesystem reliable across power failures for some hard disk types. To do this, add the barrier=1 mount option to the appropriate entry in /etc/fstab. To check if the disk drive supports this option, run **hdparm** on the applicable disk drive. For example, if:

```
hdparm -I /dev/sda | grep NCQ
```

returns non-empty output, the option is supported.

Note: Logical Volume Management (LVM) based partitions cannot use the barrier option.

8.3. Linux-3.8.1

The Linux package contains the Linux kernel.

Approximate build time:	1.0 - 5.0 SBU
Required disk space:	540 - 800 MB

8.3.1. Installation of the kernel

Building the kernel involves a few steps—configuration, compilation, and installation. Read the README file in the kernel source tree for alternative methods to the way this book configures the kernel.

Prepare for compilation by running the following command:

```
make mrproper
```

This ensures that the kernel tree is absolutely clean. The kernel team recommends that this command be issued prior to each kernel compilation. Do not rely on the source tree being clean after un-tarring.

Configure the kernel via a menu-driven interface. For general information on kernel configuration see **http://www.linuxfromscratch.org/hints/downloads/files/kernel-configuration.txt**. BLFS has some information regarding particular kernel configuration requirements of packages outside of LFS at **http://www.linuxfromscratch.org/blfs/view/svn/longindex.html#kernel-config-index**.

> **Note**
>
> Due to recent changes in udev, be sure to select:
>
> ```
> Device Drivers  --->
>   Generic Driver Options  --->
>       Maintain a devtmpfs filesystem to mount at /dev
> ```

```
make LANG=<host_LANG_value> LC_ALL= menuconfig
```

The meaning of the make parameters:

LANG=<host_LANG_value> LC_ALL=
> This establishes the locale setting to the one used on the host. This is needed for a proper menuconfig ncurses interface line drawing on UTF-8 linux text console.
> Be sure to replace *<host_LANG_value>* by the value of the $LANG variable from your host. If not set, you could use instead the host's value of $LC_ALL or $LC_CTYPE.

Alternatively, **make oldconfig** may be more appropriate in some situations. See the README file for more information.

If desired, skip kernel configuration by copying the kernel config file, .config, from the host system (assuming it is available) to the unpacked linux-3.8.1 directory. However, we do not recommend this option. It is often better to explore all the configuration menus and create the kernel configuration from scratch.

Compile the kernel image and modules:

```
make
```

If using kernel modules, module configuration in /etc/modprobe.d may be required. Information pertaining to modules and kernel configuration is located in [Section 7.4, "Device and Module Handling on an LFS System"](#) and in the kernel documentation in the linux-3.8.1/Documentation directory. Also, modprobe.conf(5) may be of interest.

Install the modules, if the kernel configuration uses them:

```
make modules_install
```

After kernel compilation is complete, additional steps are required to complete the installation. Some files need to be copied to the /boot directory.

The path to the kernel image may vary depending on the platform being used. The filename below can be changed to suit your taste, but the stem of the filename should be *vmlinuz* to be compatible with the automatic setup of the boot process described in the next section. The following command assumes an x86 architecture:

```
cp -v arch/x86/boot/bzImage /boot/vmlinuz-3.8.1-lfs-7.3
```

System.map is a symbol file for the kernel. It maps the function entry points of every function in the kernel API, as well as the addresses of the kernel data structures for the running kernel. It is used as a resource when investigating kernel problems. Issue the following command to install the map file:

```
cp -v System.map /boot/System.map-3.8.1
```

The kernel configuration file .config produced by the **make menuconfig** step above contains all the configuration selections for the kernel that was just compiled. It is a good idea to keep this file for future reference:

```
cp -v .config /boot/config-3.8.1
```

Install the documentation for the Linux kernel:

```
install -d /usr/share/doc/linux-3.8.1
cp -r Documentation/* /usr/share/doc/linux-3.8.1
```

It is important to note that the files in the kernel source directory are not owned by *root*. Whenever a package is unpacked as user *root* (like we did inside chroot), the files have the user and group IDs of whatever they were on the packager's computer. This is usually not a problem for any other package to be installed because the source tree is removed after the installation. However, the Linux source tree is often retained for a long time. Because of this, there is a chance that whatever user ID the packager used will be assigned to somebody on the machine. That person would then have write access to the kernel source.

If the kernel source tree is going to be retained, run **chown -R 0:0** on the linux-3.8.1 directory to ensure all files are owned by user *root*.

Warning

Some kernel documentation recommends creating a symlink from /usr/src/linux pointing to the kernel' source directory. This is specific to kernels prior to the 2.6 series

---

and *must not* be created on an LFS system as it can cause problems for packages you
may wish to build once your base LFS system is complete.

Warning

The headers in the system's `include` directory should *always* be the ones against which
Glibc was compiled, that is, the sanitised headers from this Linux kernel tarball.
Therefore, they should *never* be replaced by either the raw kernel headers or any other
kernel sanitized headers.

8.3.2. Configuring Linux Module Load Order

Most of the time Linux modules are loaded automatically, but sometimes it needs some specific
direction. The program that loads modules, **modprobe** or **insmod**, uses `/etc/modprobe.d/usb.conf`
for this purpose. This file needs to be created so that if the USB drivers (ehci_hcd, ohci_hcd and
uhci_hcd) have been built as modules, they will be loaded in the correct order; ehci_hcd needs to be
loaded prior to ohci_hcd and uhci_hcd in order to avoid a warning being output at boot time.

Create a new file `/etc/modprobe.d/usb.conf` by running the following:

```
install -v -m755 -d /etc/modprobe.d
cat > /etc/modprobe.d/usb.conf << "EOF"
# Begin /etc/modprobe.d/usb.conf

install ohci_hcd /sbin/modprobe ehci_hcd ; /sbin/modprobe -i ohci_hcd ; true
install uhci_hcd /sbin/modprobe ehci_hcd ; /sbin/modprobe -i uhci_hcd ; true

# End /etc/modprobe.d/usb.conf
EOF
```

8.3.3. Contents of Linux

Installed files:	config-3.8.1, vmlinux-3.8.1-lfs-7.3-3.8.1, and System.map-3.8.1
Installed directories:	/lib/modules, /usr/share/doc/linux-3.8.1

Short Descriptions

config-3.8.1	Contains all the configuration selections for the kernel
vmlinux-3.8.1-lfs-7.3	The engine of the Linux system. When turning on the computer, the kernel is the first part of the operating system that gets loaded. It detects and initializes all components of the computer's hardware, then makes these components available as a tree of files to the software and turns a single CPU into a multitasking machine capable of running scores of programs seemingly at the same time
System.map-3.8.1	A list of addresses and symbols; it maps the entry points and addresses of all the functions and data structures in the kernel

8.4. Using GRUB to Set Up the Boot Process

8.4.1. Introduction

> ### Warning
>
> Configuring GRUB incorrectly can render your system inoperable without an alternate
> boot device such as a CD-ROM. This section is not required to boot your LFS system.
> You may just want to modify your current boot loader, e.g. Grub-Legacy, GRUB2, or
> LILO.

Ensure that an emergency boot disk is ready to "rescue" the computer if the computer becomes
unusable (un-bootable). If you do not already have a boot device, you can create one. In order for the
procedure below to work, you need to jump ahead to BLFS and install **xorriso** from the **libisoburn**
package.

```
cd /tmp &&
grub-mkrescue --output=grub-img.iso &&
xorriso -as cdrecord -v dev=/dev/cdrw blank=as_needed grub-img.iso
```

8.4.2. GRUB Naming Conventions

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)*.

8.4.3. Setting Up the Configuration

GRUB works by writing data to the first physical track of the hard disk. This area is not part of any
file system. The programs there access GRUB modules in the boot partition. The default location is
/boot/grub/.

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.

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 and set up the boot track:

> ### Warning
>
> 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).

```
grub-install /dev/sda
```

8.4.4. Creating the Configuration File

Generate `/boot/grub/grub.cfg`:

```
cat > /boot/grub/grub.cfg << "EOF"
# Begin /boot/grub/grub.cfg
set default=0
set timeout=5

insmod ext2
set root=(hd0,2)

menuentry "GNU/Linux, Linux 3.8.1-lfs-7.3" {
        linux   /boot/vmlinuz-3.8.1-lfs-7.3 root=/dev/sda2 ro
}
EOF
```

> ### Note
>
> From GRUB's perspective, the kernel files are relative to the partition used. If you used a separate /boot partition, remove /boot from the above *linux* line. You will also need to change the *set root* line to point to the boot partition.

GRUB is an extremely powerful program and it provides a tremendous number of options for booting from a wide variety of devices, operating systems, and partition types. There are also many options for customization such as graphical splash screens, playing sounds, mouse input, etc. The details of these options are beyond the scope of this introduction.

> ### Caution
>
> There is a command, grub-mkconfig, that can write a configuration file automatically. It uses a set of scripts in /etc/grub.d/ and will destroy any customizations that you make. These scripts are designed primarily for non-source distributions and are not recommended for LFS. If you install a commercial Linux distribution, there is a good chance that this program will be run. Be sure to back up your grub.cfg file.

Chapter 9. The End

9.1. The End

Well done! The new LFS system is installed! We wish you much success with your shiny new custom-built Linux system.

It may be a good idea to create an /etc/lfs-release file. By having this file, it is very easy for you (and for us if you need to ask for help at some point) to find out which LFS version is installed on the system. Create this file by running:

```
echo 7.3 > /etc/lfs-release
```

It is also a good idea to create a file to show the status of your new system with resopect to the Linux Standards Base (LSB). To create this file, run:

```
cat > /etc/lsb-release << "EOF"
DISTRIB_ID="Linux From Scratch"
DISTRIB_RELEASE="7.3"
DISTRIB_CODENAME="<your name here>"
DISTRIB_DESCRIPTION="Linux From Scratch"
EOF
```

Be sure to put some sort of customization for the field 'DISTRIB_CODENAME' to make the system uniquely yours.

9.2. Get Counted

Now that you have finished the book, do you want to be counted as an LFS user? Head over to **http://www.linuxfromscratch.org/cgi-bin/lfscounter.php** and register as an LFS user by entering your name and the first LFS version you have used.

Let's reboot into LFS now.

9.3. Rebooting the System

Now that all of the software has been installed, it is time to reboot your computer. However, you should be aware of a few things. The system you have created in this book is quite minimal, and most likely will not have the functionality you would need to be able to continue forward. By installing a few extra packages from the BLFS book while still in our current chroot environment, you can leave yourself in a much better position to continue on once you reboot into your new LFS installation. Here are some suggestions:

- A text mode browser such as **Lynx** will allow you to easily view the BLFS book in one virtual terminal, while building packages in another.

- The **GPM** package will allow you to perform copy/paste actions in your virtual terminals.

- If you are in a situation where static IP configuration does not meet your networking requirements, installing a package such as **dhcpcd** or the client portion of **dhcp** may be useful.

- Installing **sudo** may be useful for building packages as a non-root user and easily installing the resulting packages in your new system.

- If you want to access your new system from a remote system within a comfortable GUI environment, install **openssh** and it's prerequsite, **openssl**.

- To make fetching files over the internet easier, install **wget**.

- If one or more of your disk drives have a GUID partition table (GPT), either **gptfdisk** or **parted** will be useful.

- Finally, a review of the following configuration files is also appropriate at this point.
  - /etc/bashrc
  - /etc/dircolors
  - /etc/fstab
  - /etc/hosts
  - /etc/inputrc
  - /etc/profile
  - /etc/resolv.conf
  - /etc/vimrc
  - /root/.bash_profile
  - /root/.bashrc
  - /etc/sysconfig/network
  - /etc/sysconfig/ifconfig.eth0

Now that we have said that, lets move on to booting our shiny new LFS installation for the first time! First exit from the chroot environment:

```
logout
```

Then unmount the virtual file systems:

```
umount -v $LFS/dev/pts

if [ -h $LFS/dev/shm ]; then
  link=$(readlink $LFS/dev/shm)
  umount -v $LFS/$link
  unset link
else
  umount -v $LFS/dev/shm
fi

umount -v $LFS/dev
umount -v $LFS/proc
umount -v $LFS/sys
```

Unmount the LFS file system itself:

```
umount -v $LFS
```

If multiple partitions were created, unmount the other partitions before unmounting the main one, like this:

```
umount -v $LFS/usr
```

```
umount -v $LFS/home
umount -v $LFS
```

Now, reboot the system with:

```
shutdown -r now
```

Assuming the GRUB boot loader was set up as outlined earlier, the menu is set to boot *LFS 7.3* automatically.

When the reboot is complete, the LFS system is ready for use and more software may be added to suit your needs.

9.4. What Now?

Thank you for reading this LFS book. We hope that you have found this book helpful and have learned more about the system creation process.

Now that the LFS system is installed, you may be wondering "What next?" To answer that question, we have compiled a list of resources for you.

- Maintenance

  Bugs and security notices are reported regularly for all software. Since an LFS system is compiled from source, it is up to you to keep abreast of such reports. There are several online resources that track such reports, some of which are shown below:

  - Freecode (**http://freecode.com/**)

    Freecode can notify you (via email) of new versions of packages installed on your system.

  - **CERT** (Computer Emergency Response Team)

    CERT has a mailing list that publishes security alerts concerning various operating systems and applications. Subscription information is available at **http://www.us-cert.gov /cas/signup.html**.

  - Bugtraq

    Bugtraq is a full-disclosure computer security mailing list. It publishes newly discovered security issues, and occasionally potential fixes for them. Subscription information is available at **http://www.securityfocus.com/archive**.

- Beyond Linux From Scratch

  The Beyond Linux From Scratch book covers installation procedures for a wide range of software beyond the scope of the LFS Book. The BLFS project is located at **http://www.linuxfromscratch.org/blfs/**.

- LFS Hints

  The LFS Hints are a collection of educational documents submitted by volunteers in the LFS community. The hints are available at **http://www.linuxfromscratch.org/hints/list.html**.

- Mailing lists

  There are several LFS mailing lists you may subscribe to if you are in need of help, want to stay current with the latest developments, want to contribute to the project, and more. See Chapter 1 - Mailing Lists for more information.

- The Linux Documentation Project

  The goal of The Linux Documentation Project (TLDP) is to collaborate on all of the issues of
  Linux documentation. The TLDP features a large collection of HOWTOs, guides, and man
  pages. It is located at **http://www.tldp.org/**.

Part IV. Appendices

Appendix A. Acronyms and Terms

ABI	Application Binary Interface
ALFS	Automated Linux From Scratch
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
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
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
PTY	pseudo terminal
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
SHA1	Secure-Hash Algorithm 1
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

Appendix B. Acknowledgments

We would like to thank the following people and organizations for their contributions to the Linux From Scratch Project.

- **Gerard Beekmans** <gerard AT linuxfromscratch D0T org> – LFS Creator, LFS Project Leader

- **Matthew Burgess** <matthew AT linuxfromscratch D0T org> – LFS Project Leader, LFS Technical Writer/Editor

- **Bruce Dubbs** <bdubbs AT linuxfromscratch D0T org> – LFS Release Manager, LFS Technical Writer/Editor

- **Jim Gifford** <jim AT linuxfromscratch D0T org> – CLFS Project Co-Leader

- **Bryan Kadzban** <bryan AT linuxfromscratch D0T org> – LFS Technical Writer

- **Randy McMurchy** <randy AT linuxfromscratch D0T org> – BLFS Project Leader, LFS Editor

- **DJ Lucas** <dj AT linuxfromscratch D0T org> – LFS and BLFS Editor

- **Ken Moffat** <ken AT linuxfromscratch D0T org> – LFS and CLFS Editor

- **Ryan Oliver** <ryan AT linuxfromscratch D0T 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 AT macana-es D0T com> – Spanish LFS translation project

- **Johan Lenglet** <johan AT linuxfromscratch D0T org> – French LFS translation project

- **Anderson Lizardo** <lizardo AT linuxfromscratch D0T org> – Portuguese LFS translation project

- **Thomas Reitelbach** <tr AT erdfunkstelle D0T de> – German LFS translation project

Mirror Maintainers

North American Mirrors

- **Scott Kveton** <scott AT osuosl D0T org> – lfs.oregonstate.edu mirror

- **William Astle** <lost AT l-w D0T 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 AT linuxfromscratch D0T org> – lfsmirror.lfs-es.info mirror

- **Luis Falcon** <Luis Falcon> – torredehanoi.org mirror

European Mirrors

- **Guido Passet** <guido AT primerelay D0T net> – nl.linuxfromscratch.org mirror

- **Bastiaan Jacques** <baafie AT planet D0T nl> – lfs.pagefault.net mirror

- **Sven Cranshoff** <sven D0T cranshoff AT lineo D0T be> – lfs.lineo.be mirror

- Scarlet Belgium – lfs.scarlet.be mirror

- **Sebastian Faulborn** <info AT aliensoft D0T org> – lfs.aliensoft.org mirror

- **Stuart Fox** <stuart AT dontuse D0T ms> – lfs.dontuse.ms mirror

- **Ralf Uhlemann** <admin AT realhost D0T de> – lfs.oss-mirror.org mirror

- **Antonin Sprinzl** <Antonin D0T Sprinzl AT tuwien D0T ac D0T at> – at.linuxfromscratch.org mirror

- **Fredrik Danerklint** <fredan-lfs AT fredan D0T org> – se.linuxfromscratch.org mirror

- **Franck** <franck AT linuxpourtous D0T com> – lfs.linuxpourtous.com mirror

- **Philippe Baqué** <baque AT cict D0T fr> – lfs.cict.fr mirror

- **Vitaly Chekasin** <gyouja AT pilgrims D0T ru> – lfs.pilgrims.ru mirror

- **Benjamin Heil** <kontakt AT wankoo D0T org> – lfs.wankoo.org mirror

Asian Mirrors

- **Satit Phermsawang** <satit AT wbac D0T ac D0T th> – lfs.phayoune.org mirror

- **Shizunet Co.,Ltd.** <info AT shizu-net D0T jp> – lfs.mirror.shizu-net.jp mirror

- **Init World** <http://www.initworld.com/> – lfs.initworld.com mirror

Australian Mirrors

- **Jason Andrade** <jason AT dstc D0T edu D0T au> – au.linuxfromscratch.org mirror

Former Project Team Members

- **Christine Barczak** <theladyskye AT linuxfromscratch D0T 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 AT linuxfromscratch D0T org> – LFS-Bootscripts Maintainer

- Timothy Bauscher

- Robert Briggs

- Ian Chilton

- **Jeroen Coumans** <jeroen AT linuxfromscratch D0T org> – Website Developer, FAQ Maintainer

- **Manuel Canales Esparcia** <manuel AT linuxfromscratch D0T org> – LFS/BLFS/HLFS XML and XSL Maintainer

- Alex Groenewoud – LFS Technical Writer

- Marc Heerdink

- **Jeremy Huntwork** <jhuntwork AT linuxfromscratch D0T org> – LFS Technical Writer, LFS LiveCD Maintainer

- Mark Hymers

- Seth W. Klein – FAQ maintainer

- **Nicholas Leippe** <nicholas AT linuxfromscratch D0T org> – Wiki Maintainer

- **Anderson Lizardo** <lizardo AT linuxfromscratch D0T org> – Website Backend-Scripts Maintainer

- **Dan Nicholson** <dnicholson AT linuxfromscratch D0T org> – LFS and BLFS Editor

- **Alexander E. Patrakov** <alexander AT linuxfromscratch D0T org> – LFS Technical Writer, LFS Internationalization Editor, LFS Live CD Maintainer

- Simon Perreault

- **Scot Mc Pherson** <scot AT linuxfromscratch D0T org> – LFS NNTP Gateway Maintainer

- **Greg Schafer** <gschafer AT zip D0T com D0T 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 AT linuxfromscratch D0T org> – Bugzilla Maintainer

- **Tushar Teredesai** <tushar AT linuxfromscratch D0T org> – BLFS Book Editor, Hints and Patches Project Leader

- **Jeremy Utley** <jeremy AT linuxfromscratch D0T org> – LFS Technical Writer, Bugzilla Maintainer, LFS-Bootscripts Maintainer

- **Zack Winkles** <zwinkles AT gmail D0T com> – LFS Technical Writer

Appendix 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 recommended 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

Installation depends on:	Bash, Coreutils, Grep, M4, Make, Perl, Sed, and Texinfo
Test suite depends on:	Automake, Diffutils, Findutils, GCC, and Libtool
Must be installed before:	Automake
Optional dependencies:	Emacs

Automake

Installation depends on:	Autoconf, Bash, Coreutils, Gettext, Grep, M4, Make, Perl, Sed, and Texinfo
Test suite depends on:	Binutils, Bison, Bzip2, DejaGNU, Diffutils, Expect, Findutils, Flex, GCC, Gettext, Gzip, Libtool, and Tar.
Must be installed before:	None
Optional dependencies:	None

Bash

Installation depends on:	Bash, Binutils, Bison, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Make, Ncurses, Patch, Readline, Sed, and Texinfo
Test suite depends on:	Shadow
Must be installed before:	None
Optional dependencies:	Xorg

Binutils

Installation depends on:	Bash, Binutils, Coreutils, Diffutils, File, Gawk, GCC, Glibc, Grep, Make, Perl, Sed, Texinfo and Zlib
Test suite depends on:	DejaGNU and Expect
Must be installed before:	None
Optional dependencies:	None

Bison

Installation depends on:	Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, M4, Make, and Sed
Test suite depends on:	Diffutils and Findutils
Must be installed before:	Flex, Kbd, and Tar
Optional dependencies:	Doxygen (test suite)

Bzip2

Installation depends on:	Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Make, and Patch
Test suite depends on:	None
Must be installed before:	None
Optional dependencies:	None

Coreutils

Installation depends on:	Bash, Binutils, Coreutils, GCC, Gettext, Glibc, GMP, Grep, Make, Patch, Perl, Sed, and Texinfo
Test suite depends on:	Diffutils, E2fsprogs, Findutils, Shadow, and Util-linux
Must be installed before:	Bash, Diffutils, Findutils, Man-DB, and Udev
Optional dependencies:	Perl Expect and IO:Tty modules (for test suite)

DejaGNU

Installation depends on:	Bash, Coreutils, Diffutils, GCC, Grep, Make, and Sed
Test suite depends on:	No test suite available
Must be installed before:	None
Optional dependencies:	None

Diffutils

Installation depends on:	Bash, Binutils, Coreutils, Gawk, GCC, Gettext, Glibc, Grep, Make, Sed, and Texinfo
Test suite depends on:	Diffutils, Perl
Must be installed before:	None
Optional dependencies:	None

Expect

Installation depends on:	Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Grep, Make, Patch, Sed, and Tcl
Test suite depends on:	None
Must be installed before:	None
Optional dependencies:	None

E2fsprogs

Installation depends on:	Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Gzip, Make, Sed, Texinfo, and Util-linux
Test suite depends on:	Procps-ng,Psmisc
Must be installed before:	None
Optional dependencies:	None

File

Installation depends on:	Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Make, Sed, and Zlib
Test suite depends on:	None
Must be installed before:	None
Optional dependencies:	None

Findutils

Installation depends on:	Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, Make, Sed, and Texinfo
Test suite depends on:	DejaGNU, Diffutils, and Expect
Must be installed before:	None
Optional dependencies:	None

Flex

Installation depends on:	Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, M4, Make, Patch, Sed, and Texinfo
Test suite depends on:	Bison and Gawk
Must be installed before:	IPRoute2, Kbd, and Man-DB
Optional dependencies:	None

Gawk

Installation depends on:	Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, Make, Patch, Sed and, Texinfo
Test suite depends on:	Diffutils
Must be installed before:	None
Optional dependencies:	None

Gcc

Installation depends on:	Bash, Binutils, Coreutils, Diffutils, Findutils, Gawk, GCC, Gettext, Glibc, GMP, Grep, M4, Make, MPC, MPFR, Patch, Perl, Sed, Tar, and Texinfo
Test suite depends on:	DejaGNU and Expect
Must be installed before:	None
Optional dependencies:	**CLooG-PPL**, **GNAT** and **PPL**

GDBM

Installation depends on:	Bash, Binutils, Coreutils, Diffutils, GCC, Grep, Make, and Sed
Test suite depends on:	None
Must be installed before:	None
Optional dependencies:	None

Gettext

Installation depends on:	Bash, Binutils, Coreutils, Gawk, GCC, Glibc, Grep, Make, Sed, and Texinfo
Test suite depends on:	Diffutils, Perl, and Tcl
Must be installed before:	Automake
Optional dependencies:	None

Glibc

Installation depends on:	Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Gettext, Grep, Gzip, Linux API Headers, Make, Perl, Sed, and Texinfo
Test suite depends on:	File
Must be installed before:	None
Optional dependencies:	None

GMP

Installation depends on:	Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, M4, Make, Sed and Texinfo
Test suite depends on:	None
Must be installed before:	MPFR, GCC
Optional dependencies:	None

Grep

Installation depends on:	Bash, Binutils, Coreutils, Diffutils, GCC, Gettext, Glibc, Grep, Make, Patch, Sed, and Texinfo
Test suite depends on:	Gawk
Must be installed before:	Man-DB
Optional dependencies:	Pcre, Xorg, and CUPS

Groff

Installation depends on:	Bash, Binutils, Bison, Coreutils, Gawk, GCC, Glibc, Grep, Make, Patch, Sed, and Texinfo
Test suite depends on:	No test suite available
Must be installed before:	Man-DB and Perl
Optional dependencies:	GPL Ghostscript

GRUB

Installation depends on:	Bash, Binutils, Bison, Coreutils, Diffutils, GCC, Gettext, Glibc, Grep, Make, Ncurses, Sed, Texinfo, and Xz
Test suite depends on:	None
Must be installed before:	None
Optional dependencies:	None

Gzip

Installation depends on:	Bash, Binutils, Coreutils, GCC, Glibc, Grep, Make, Sed, and Texinfo
Test suite depends on:	Diffutils. Less
Must be installed before:	Man-DB
Optional dependencies:	None

Iana-Etc

Installation depends on:	Coreutils, Gawk, and Make
Test suite depends on:	No test suite available
Must be installed before:	Perl
Optional dependencies:	None

Inetutils

Installation depends on:	Bash, Binutils, Coreutils, GCC, Glibc, Grep, Make, Ncurses, Patch, Sed, Texinfo, and Zlib
Test suite depends on:	No test suite available
Must be installed before:	Tar
Optional dependencies:	None

IProute2

Installation depends on:	Bash, Bison, Coreutils, Flex, GCC, Glibc, Make, and Linux API Headers
Test suite depends on:	No test suite available
Must be installed before:	None
Optional dependencies:	None

Kbd

Installation depends on:	Bash, Binutils, Bison, Coreutils, Flex, GCC, Gettext, Glibc, Gzip, Make, Patch, and Sed
Test suite depends on:	No test suite available
Must be installed before:	None
Optional dependencies:	None

Kmod

Installation depends on:	Bash, Binutils, Bison, Coreutils, Flex, GCC, Gettext, Glibc, Gzip, Make, Sed, Xz-Utils, Zlib
Test suite depends on:	No test suite available
Must be installed before:	Udev
Optional dependencies:	None

Less

Installation depends on:	Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Grep, Make, Ncurses, and Sed
Test suite depends on:	No test suite available
Must be installed before:	Gzip
Optional dependencies:	Pcre

Libpipeline

Installation depends on:	Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Make,

	Sed, and Texinfo
Test suite depends on:	No test suite available
Must be installed before:	Man-DB
Optional dependencies:	None

Libtool

Installation depends on:	Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Make, Sed, and Texinfo
Test suite depends on:	Findutils
Must be installed before:	None
Optional dependencies:	None

Linux Kernel

Installation depends on:	Bash, Binutils, Coreutils, Diffutils, Findutils, GCC, Glibc, Grep, Gzip, Kmod, Make, Ncurses, Perl, and Sed
Test suite depends on:	No test suite available
Must be installed before:	None
Optional dependencies:	None

M4

Installation depends on:	Bash, Binutils, Coreutils, GCC, Glibc, Grep, Make, Sed, and Texinfo
Test suite depends on:	Diffutils
Must be installed before:	Autoconf and Bison
Optional dependencies:	libsigsegv

Make

Installation depends on:	Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, Make, Sed, and Texinfo
Test suite depends on:	Perl and Procps-ng
Must be installed before:	None
Optional dependencies:	None

Man-DB

Installation depends on:	Bash, Binutils, Bzip2, Coreutils, Flex, GCC, GDBM, Gettext, Glibc, Grep, Groff, Gzip, Less, Libpipeline, Make, Sed, and Xz
Test suite depends on:	Not run. Requires Man-DB test suite package
Must be installed before:	None
Optional dependencies:	None

Man-Pages

Installation depends on:	Bash, Coreutils, and Make
Test suite depends on:	No test suite available
Must be installed before:	None
Optional dependencies:	None

MPC

Installation depends on:	Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, GMP, Make, MPFR, Sed and Texinfo
Test suite depends on:	None
Must be installed before:	GCC
Optional dependencies:	None

MPFR

Installation depends on:	Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, GMP, Make, Sed and Texinfo

Test suite depends on:	None
Must be installed before:	GCC
Optional dependencies:	None

Ncurses

Installation depends on:	Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Make, Patch, and Sed
Test suite depends on:	No test suite available
Must be installed before:	Bash, GRUB, Inetutils, Less, Procps-ng, Psmisc, Readline, Texinfo, Util-linux, and Vim
Optional dependencies:	None

Patch

Installation depends on:	Bash, Binutils, Coreutils, GCC, Glibc, Grep, Make, and Sed
Test suite depends on:	No test suite available
Must be installed before:	None
Optional dependencies:	Ed

Perl

Installation depends on:	Bash, Binutils, Coreutils, Gawk, GCC, GDBM, Glibc, Grep, Groff, Make, Sed, and Zlib
Test suite depends on:	Iana-Etc and Procps-ng
Must be installed before:	Autoconf
Optional dependencies:	None

Pkg-config

Installation depends on:	Bash, Binutils, Coreutils, Gawk, GCC, Glibc, Grep, Make, Popt, and Sed
Test suite depends on:	None
Must be installed before:	Kmod
Optional dependencies:	None

Popt

Installation depends on:	Bash, Binutils, Coreutils, Gawk, GCC, Glibc, Grep, Make
Test suite depends on:	Diffutils and Sed
Must be installed before:	Pkg-config
Optional dependencies:	None

Procps-ng

Installation depends on:	Bash, Binutils, Coreutils, GCC, Glibc, Make, and Ncurses
Test suite depends on:	DejaGNU
Must be installed before:	None
Optional dependencies:	None

Psmisc

Installation depends on:	Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, Make, Ncurses, and Sed
Test suite depends on:	No test suite available
Must be installed before:	None
Optional dependencies:	None

Readline

Installation depends on:	Bash, Binutils, Coreutils, Gawk, GCC, Glibc, Grep, Make, Ncurses, Patch, Sed, and Texinfo
Test suite depends on:	No test suite available

Must be installed before:	Bash
Optional dependencies:	None

Sed

Installation depends on:	Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, Make, Sed, and Texinfo
Test suite depends on:	Diffutils and Gawk
Must be installed before:	E2fsprogs, File, Libtool, and Shadow
Optional dependencies:	Cracklib

Shadow

Installation depends on:	Bash, Binutils, Coreutils, Diffutils, Findutils, Gawk, GCC, Gettext, Glibc, Grep, Make, and Sed
Test suite depends on:	No test suite available
Must be installed before:	Coreutils
Optional dependencies:	Acl, Attr, Cracklib, PAM

Sysklogd

Installation depends on:	Binutils, Coreutils, GCC, Glibc, Make, and Patch
Test suite depends on:	No test suite available
Must be installed before:	None
Optional dependencies:	None

Sysvinit

Installation depends on:	Binutils, Coreutils, GCC, Glibc, Make, and Sed
Test suite depends on:	No test suite available
Must be installed before:	None
Optional dependencies:	None

Tar

Installation depends on:	Bash, Binutils, Bison, Coreutils, GCC, Gettext, Glibc, Grep, Inetutils, Make, Sed, and Texinfo
Test suite depends on:	Autoconf, Diffutils, Findutils, Gawk, and Gzip
Must be installed before:	None
Optional dependencies:	None

Tcl

Installation depends on:	Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Grep, Make, and Sed
Test suite depends on:	None
Must be installed before:	None
Optional dependencies:	None

Texinfo

Installation depends on:	Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, Make, Ncurses, Patch, and Sed
Test suite depends on:	None
Must be installed before:	None
Optional dependencies:	None

Udev

Installation depends on:	Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Kmod, Make, and Sed
Test suite depends on:	No test suite available
Must be installed before:	None
Optional dependencies:	Glib, Pci-Utils, Python, Systemd, USB-Utils

Util-linux

Installation depends on:	Bash, Binutils, Coreutils, Diffutils, Findutils, Gawk, GCC, Gettext, Glibc, Grep, Make, Ncurses, Sed, and Zlib
Test suite depends on:	No test suite available
Must be installed before:	None
Optional dependencies:	None

Vim

Installation depends on:	Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Grep, Make, Ncurses, and Sed
Test suite depends on:	None
Must be installed before:	None
Optional dependencies:	Xorg, GTK+2, LessTif, Python, Tcl, Ruby, and GPM

Xz

Installation depends on:	Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, and Make.
Test suite depends on:	None
Must be installed before:	GRUB, Kmod, Man-DB, Udev
Optional dependencies:	None

Zlib

Installation depends on:	Bash, Binutils, Coreutils, GCC, Glibc, Make, and Sed
Test suite depends on:	None
Must be installed before:	File, Kmod, Perl, and Util-linux
Optional dependencies:	None

Appendix D. Boot and sysconfig scripts version-20130123

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/bash
########################################################################
# Begin rc
#
# Description : Main Run Level Control Script
#
# Authors     : Gerard Beekmans  - gerard AT linuxfromscratch D0T org
#             : DJ Lucas - dj AT linuxfromscratch D0T org
# Update      : Bruce Dubbs - bdubbs AT linuxfromscratch D0T org
#
# Version     : LFS 7.0
#
########################################################################

. /lib/lsb/init-functions
```

```
print_error_msg()
{
    log_failure_msg
    # $i is set when called
    MSG="FAILURE:\n\nYou should not be reading this error message.\n\n"
    MSG="${MSG}It means that an unforeseen error took place in\n"
    MSG="${MSG}${i},\n"
    MSG="${MSG}which exited with a return value of ${error_value}.\n"

    MSG="${MSG}If you're able to track this error down to a bug in one of\n"
    MSG="${MSG}the files provided by the files provided by\n"
    MSG="${MSG}the ${DISTRO_MINI} book, please be so kind to inform us at\n"
    MSG="${MSG}${DISTRO_CONTACT}.\n"
    log_failure_msg "${MSG}"

    log_info_msg "Press Enter to continue..."
    wait_for_user
}

check_script_status()
{
    # $i is set when called
    if [ ! -f ${i} ]; then
        log_warning_msg "${i} is not a valid symlink."
        continue
    fi

    if [ ! -x ${i} ]; then
        log_warning_msg "${i} is not executable, skipping."
        continue
    fi
}

run()
{
    if [ -z $interactive ]; then
        ${1} ${2}
        return $?
    fi

    while true; do
        read -p "Run ${1} ${2} (Yes/no/continue)? " -n 1 runit
        echo

        case ${runit} in
            c | C)
                interactive=""
                ${i} ${2}
                ret=${?}
                break;
                ;;

            n | N)
                return 0
                ;;
```

```
            y | Y)
                ${i} ${2}
                ret=${?}
                break
                ;;
        esac
    done

    return $ret
}

# Read any local settings/overrides
[ -r /etc/sysconfig/rc.site ] && source /etc/sysconfig/rc.site

DISTRO=${DISTRO:-"Linux From Scratch"}
DISTRO_CONTACT=${DISTRO_CONTACT:-"lfs-dev@linuxfromscratch.org (Registration required
DISTRO_MINI=${DISTRO_MINI:-"LFS"}
IPROMPT=${IPROMPT:-"no"}

# These 3 signals will not cause our script to exit
trap "" INT QUIT TSTP

[ "${1}" != "" ] && runlevel=${1}

if [ "${runlevel}" == "" ]; then
    echo "Usage: ${0} <runlevel>" >&2
    exit 1
fi

previous=${PREVLEVEL}
[ "${previous}" == "" ] && previous=N

if [ ! -d /etc/rc.d/rc${runlevel}.d ]; then
    log_info_msg "/etc/rc.d/rc${runlevel}.d does not exist.\n"
    exit 1
fi

if [ "$runlevel" == "6" -o "$runlevel" == "0" ]; then IPROMPT="no"; fi

# Note: In ${LOGLEVEL:-7}, it is ':' 'dash' '7', not minus 7
if [ "$runlevel" == "S" ]; then
    [ -r /etc/sysconfig/console ] && source /etc/sysconfig/console
    dmesg -n "${LOGLEVEL:-7}"
fi

if [ "${IPROMPT}" == "yes" -a "${runlevel}" == "S" ]; then
    # The total length of the distro welcome string, without escape codes
    wlen=${wlen:-$(echo "Welcome to ${DISTRO}" | wc -c )}
    welcome_message=${welcome_message:-"Welcome to ${INFO}${DISTRO}${NORMAL}"}

    # The total length of the interactive string, without escape codes
    ilen=${ilen:-$(echo "Press 'I' to enter interactive startup" | wc -c )}
    i_message=${i_message:-"Press '${FAILURE}I${NORMAL}' to enter interactive startup'

    # dcol and icol are spaces before the message to center the message
    # on screen. itime is the amount of wait time for the user to press a key
```

```
    wcol=$(( ( ${COLUMNS} - ${wlen} ) / 2 ))
    icol=$(( ( ${COLUMNS} - ${ilen} ) / 2 ))
    itime=${itime:-"3"}

    echo -e "\n\n"
    echo -e "\\033[${wcol}G${welcome_message}"
    echo -e "\\033[${icol}G${i_message}${NORMAL}"
    echo ""
    read -t "${itime}" -n 1 interactive 2>&1 > /dev/null
fi

# Make lower case
[ "${interactive}" == "I" ] && interactive="i"
[ "${interactive}" != "i" ] && interactive=""

# Read the state file if it exists from runlevel S
[ -r /var/run/interactive ] && source /var/run/interactive

# Attempt to stop all services started by the previous runlevel,
# and killed in this runlevel
if [ "${previous}" != "N" ]; then
    for i in $(ls -v /etc/rc.d/rc${runlevel}.d/K* 2> /dev/null)
    do
        check_script_status

        suffix=${i#/etc/rc.d/rc$runlevel.d/K[0-9][0-9]}
        prev_start=/etc/rc.d/rc$previous.d/S[0-9][0-9]$suffix
        sysinit_start=/etc/rc.d/rcS.d/S[0-9][0-9]$suffix

        if [ "${runlevel}" != "0" -a "${runlevel}" != "6" ]; then
            if [ ! -f ${prev_start} -a  ! -f ${sysinit_start} ]; then
                MSG="WARNING:\n\n${i} can't be "
                MSG="${MSG}executed because it was not "
                MSG="${MSG}not started in the previous "
                MSG="${MSG}runlevel (${previous})."
                log_warning_msg "$MSG"
                continue
            fi
        fi

        run ${i} stop
        error_value=${?}

        if [ "${error_value}" != "0" ]; then print_error_msg; fi
    done
fi

if [ "${previous}" == "N" ]; then export IN_BOOT=1; fi

if [ "$runlevel" == "6" -a -n "${FASTBOOT}" ]; then
    touch /fastboot
fi


# Start all functions in this runlevel
for i in $( ls -v /etc/rc.d/rc${runlevel}.d/S* 2> /dev/null)
do
```

```
    if [ "${previous}" != "N" ]; then
        suffix=${i#/etc/rc.d/rc$runlevel.d/S[0-9][0-9]}
        stop=/etc/rc.d/rc$runlevel.d/K[0-9][0-9]$suffix
        prev_start=/etc/rc.d/rc$previous.d/S[0-9][0-9]$suffix

        [ -f ${prev_start} -a ! -f ${stop} ] && continue
    fi

    check_script_status

    case ${runlevel} in
        0|6)
            run ${i} stop
            ;;
        *)
            run ${i} start
            ;;
    esac

    error_value=${?}

    if [ "${error_value}" != "0" ]; then print_error_msg; fi
done

# Store interactive variable on switch from runlevel S and remove if not
if [ "${runlevel}" == "S" -a "${interactive}" == "i" ]; then
     echo "interactive=\"i\"" > /var/run/interactive
else
     rm -f /var/run/interactive 2> /dev/null
fi

# Copy the boot log on initial boot only
if [ "${previous}" == "N" -a  "${runlevel}" != "S" ]; then
    cat /run/var/bootlog >> /var/log/boot.log

    # Mark the end of boot
    echo "--------" >> /var/log/boot.log

    # Remove the temporary file
    rm -f /run/var/bootlog 2> /dev/null
fi

# End rc
```

D.2. /lib/lsb/init-functions

```
#!/bin/sh
########################################################################
#
# Begin /lib/lsb/init-funtions
#
# Description : Run Level Control Functions
#
# Authors     : Gerard Beekmans - gerard AT linuxfromscratch D0T org
#             : DJ Lucas - dj AT linuxfromscratch D0T org
# Update      : Bruce Dubbs - bdubbs AT linuxfromscratch D0T org
```

```
#
# Version     : LFS 7.0
#
# Notes       : With code based on Matthias Benkmann's simpleinit-msb
#                 http://winterdrache.de/linux/newboot/index.html
#
#                 The file should be located in /lib/lsb
#
################################################################################

## Environmental setup
# Setup default values for environment
umask 022
export PATH="/bin:/usr/bin:/sbin:/usr/sbin"

## Screen Dimensions
# Find current screen size
if [ -z "${COLUMNS}" ]; then
    COLUMNS=$(stty size)
    COLUMNS=${COLUMNS##* }
fi

# When using remote connections, such as a serial port, stty size returns 0
if [ "${COLUMNS}" = "0" ]; then
    COLUMNS=80
fi

## Measurements for positioning result messages
COL=$((${COLUMNS} - 8))
WCOL=$((${COL} - 2))

## Set Cursor Position Commands, used via echo
SET_COL="\\033[${COL}G"      # at the $COL char
SET_WCOL="\\033[${WCOL}G"    # at the $WCOL char
CURS_UP="\\033[1A\\033[0G"   # Up one line, at the 0'th char
CURS_ZERO="\\033[0G"

## Set color commands, used via echo
# Please consult `man console_codes for more information
# under the "ECMA-48 Set Graphics Rendition" section
#
# Warning: when switching from a 8bit to a 9bit font,
# the linux console will reinterpret the bold (1;) to
# the top 256 glyphs of the 9bit font.  This does
# not affect framebuffer consoles

NORMAL="\\033[0;39m"         # Standard console grey
SUCCESS="\\033[1;32m"        # Success is green
WARNING="\\033[1;33m"        # Warnings are yellow
FAILURE="\\033[1;31m"        # Failures are red
INFO="\\033[1;36m"           # Information is light cyan
BRACKET="\\033[1;34m"        # Brackets are blue

# Use a colored prefix
BMPREFIX="     "
SUCCESS_PREFIX="${SUCCESS}  *  ${NORMAL}"
FAILURE_PREFIX="${FAILURE}*****${NORMAL}"
```

```
WARNING_PREFIX="${WARNING} *** ${NORMAL}"

SUCCESS_SUFFIX="${BRACKET}[${SUCCESS}  OK  ${BRACKET}]${NORMAL}"
FAILURE_SUFFIX="${BRACKET}[${FAILURE} FAIL ${BRACKET}]${NORMAL}"
WARNING_SUFFIX="${BRACKET}[${WARNING} WARN ${BRACKET}]${NORMAL}"

BOOTLOG=/run/var/bootlog
KILLDELAY=3

# Set any user specified environment variables e.g. HEADLESS
[ -r /etc/sysconfig/rc.site ]  && . /etc/sysconfig/rc.site

################################################################################
# start_daemon()                                                               #
# Usage: start_daemon [-f] [-n nicelevel] [-p pidfile] pathname [args...]      #
#                                                                              #
# Purpose: This runs the specified program as a daemon                         #
#                                                                              #
# Inputs: -f: (force) run the program even if it is already running.           #
#         -n nicelevel: specify a nice level. See 'man nice(1)'.               #
#         -p pidfile: use the specified file to determine PIDs.                #
#         pathname: the complete path to the specified program                 #
#         args: additional arguments passed to the program (pathname)          #
#                                                                              #
# Return values (as defined by LSB exit codes):                                #
#       0 - program is running or service is OK                                #
#       1 - generic or unspecified error                                       #
#       2 - invalid or excessive argument(s)                                   #
#       5 - program is not installed                                           #
################################################################################
start_daemon()
{
    local force=""
    local nice="0"
    local pidfile=""
    local pidlist=""
    local retval=""

    # Process arguments
    while true
    do
        case "${1}" in

            -f)
                force="1"
                shift 1
                ;;

            -n)
                nice="${2}"
                shift 2
                ;;

            -p)
                pidfile="${2}"
                shift 2
                ;;
```

```
            -*)
                return 2
                ;;

            *)
                program="${1}"
                break
                ;;
        esac
    done

    # Check for a valid program
    if [ ! -e "${program}" ]; then return 5; fi

    # Execute
    if [ -z "${force}" ]; then
        if [ -z "${pidfile}" ]; then
            # Determine the pid by discovery
            pidlist=`pidofproc "${1}"`
            retval="${?}"
        else
            # The PID file contains the needed PIDs
            # Note that by LSB requirement, the path must be given to pidofproc,
            # however, it is not used by the current implementation or standard.
            pidlist=`pidofproc -p "${pidfile}" "${1}"`
            retval="${?}"
        fi

        # Return a value ONLY
        # It is the init script's (or distribution's functions) responsibilty
        # to log messages!
        case "${retval}" in

            0)
                # Program is already running correctly, this is a
                # successful start.
                return 0
                ;;

            1)
                # Program is not running, but an invalid pid file exists
                # remove the pid file and continue
                rm -f "${pidfile}"
                ;;

            3)
                # Program is not running and no pidfile exists
                # do nothing here, let start_deamon continue.
                ;;

            *)
                # Others as returned by status values shall not be interpreted
                # and returned as an unspecified error.
                return 1
                ;;
        esac
```

```
        fi

        # Do the start!
        nice -n "${nice}" "${@}"
}


################################################################################
# killproc()                                                                   #
# Usage: killproc [-p pidfile] pathname [signal]                               #
#                                                                              #
# Purpose: Send control signals to running processes                           #
#                                                                              #
# Inputs: -p pidfile, uses the specified pidfile                               #
#         pathname, pathname to the specified program                          #
#         signal, send this signal to pathname                                 #
#                                                                              #
# Return values (as defined by LSB exit codes):                                #
#       0 - program (pathname) has stopped/is already stopped or a             #
#           running program has been sent specified signal and stopped         #
#           successfully                                                       #
#       1 - generic or unspecified error                                       #
#       2 - invalid or excessive argument(s)                                   #
#       5 - program is not installed                                           #
#       7 - program is not running and a signal was supplied                   #
################################################################################
killproc()
{
    local pidfile
    local program
    local prefix
    local progname
    local signal="-TERM"
    local fallback="-KILL"
    local nosig
    local pidlist
    local retval
    local pid
    local delay="30"
    local piddead
    local dtime

    # Process arguments
    while true; do
        case "${1}" in
            -p)
                pidfile="${2}"
                shift 2
                ;;

            *)
                program="${1}"
                if [ -n "${2}" ]; then
                    signal="${2}"
                    fallback=""
                else
                    nosig=1
                fi
```

```
                        # Error on additional arguments
                        if [ -n "${3}" ]; then
                            return 2
                        else
                            break
                        fi
                        ;;
            esac
        done

        # Check for a valid program
        if [ ! -e "${program}" ]; then return 5; fi

        # Check for a valid signal
        check_signal "${signal}"
        if [ "${?}" -ne "0" ]; then return 2; fi

        # Get a list of pids
        if [ -z "${pidfile}" ]; then
            # determine the pid by discovery
            pidlist=`pidofproc "${1}"`
            retval="${?}"
        else
            # The PID file contains the needed PIDs
            # Note that by LSB requirement, the path must be given to pidofproc,
            # however, it is not used by the current implementation or standard.
            pidlist=`pidofproc -p "${pidfile}" "${1}"`
            retval="${?}"
        fi

        # Return a value ONLY
        # It is the init script's (or distribution's functions) responsibilty
        # to log messages!
        case "${retval}" in

            0)
                # Program is running correctly
                # Do nothing here, let killproc continue.
                ;;

            1)
                # Program is not running, but an invalid pid file exists
                # Remove the pid file.
                rm -f "${pidfile}"

                # This is only a success if no signal was passed.
                if [ -n "${nosig}" ]; then
                    return 0
                else
                    return 7
                fi
                ;;

            3)
                # Program is not running and no pidfile exists
                # This is only a success if no signal was passed.
```

```
            if [ -n "${nosig}" ]; then
                return 0
            else
                return 7
            fi
            ;;

    *)
            # Others as returned by status values shall not be interpreted
            # and returned as an unspecified error.
            return 1
            ;;
esac

# Perform different actions for exit signals and control signals
check_sig_type "${signal}"

if [ "${?}" -eq "0" ]; then # Signal is used to terminate the program

    # Account for empty pidlist (pid file still exists and no
    # signal was given)
    if [ "${pidlist}" != "" ]; then

        # Kill the list of pids
        for pid in ${pidlist}; do

            kill -0 "${pid}" 2> /dev/null

            if [ "${?}" -ne "0" ]; then
                # Process is dead, continue to next and assume all is well
                continue
            else
                kill "${signal}" "${pid}" 2> /dev/null

                # Wait up to ${delay}/10 seconds to for "${pid}" to
                # terminate in 10ths of a second

                while [ "${delay}" -ne "0" ]; do
                    kill -0 "${pid}" 2> /dev/null || piddead="1"
                    if [ "${piddead}" = "1" ]; then break; fi
                    sleep 0.1
                    delay="$(( ${delay} - 1 ))"
                done

                # If a fallback is set, and program is still running, then
                # use the fallback
                if [ -n "${fallback}" -a "${piddead}" != "1" ]; then
                    kill "${fallback}" "${pid}" 2> /dev/null
                    sleep 1
                    # Check again, and fail if still running
                    kill -0 "${pid}" 2> /dev/null && return 1
                fi
            fi
        done
    fi

    # Check for and remove stale PID files.
```

```
            if [ -z "${pidfile}" ]; then
                # Find the basename of $program
                prefix=`echo "${program}" | sed 's/[^/]*$//'`
                progname=`echo "${program}" | sed "s@${prefix}@@"`

                if [ -e "/var/run/${progname}.pid" ]; then
                    rm -f "/var/run/${progname}.pid" 2> /dev/null
                fi
            else
                if [ -e "${pidfile}" ]; then rm -f "${pidfile}" 2> /dev/null; fi
            fi

        # For signals that do not expect a program to exit, simply
        # let kill do it's job, and evaluate kills return for value

        else # check_sig_type - signal is not used to terminate program
            for pid in ${pidlist}; do
                kill "${signal}" "${pid}"
                if [ "${?}" -ne "0" ]; then return 1; fi
            done
        fi
}

################################################################################
# pidofproc()                                                                  #
# Usage: pidofproc [-p pidfile] pathname                                       #
#                                                                              #
# Purpose: This function returns one or more pid(s) for a particular daemon    #
#                                                                              #
# Inputs: -p pidfile, use the specified pidfile instead of pidof              #
#         pathname, path to the specified program                             #
#                                                                              #
# Return values (as defined by LSB status codes):                             #
#       0 - Success (PIDs to stdout)                                          #
#       1 - Program is dead, PID file still exists (remaining PIDs output)    #
#       3 - Program is not running (no output)                               #
################################################################################
pidofproc()
{
    local pidfile
    local program
    local prefix
    local progname
    local pidlist
    local lpids
    local exitstatus="0"

    # Process arguments
    while true; do
        case "${1}" in

            -p)
                pidfile="${2}"
                shift 2
                ;;

            *)
```

```
                program="${1}"
                if [ -n "${2}" ]; then
                    # Too many arguments
                    # Since this is status, return unknown
                    return 4
                else
                    break
                fi
                ;;
        esac
    done

    # If a PID file is not specified, try and find one.
    if [ -z "${pidfile}" ]; then
        # Get the program's basename
        prefix=`echo "${program}" | sed 's/[^/]*$//'`

        if [ -z "${prefix}" ]; then
            progname="${program}"
        else
            progname=`echo "${program}" | sed "s@${prefix}@@"`
        fi

        # If a PID file exists with that name, assume that is it.
        if [ -e "/var/run/${progname}.pid" ]; then
            pidfile="/var/run/${progname}.pid"
        fi
    fi

    # If a PID file is set and exists, use it.
    if [ -n "${pidfile}" -a -e "${pidfile}" ]; then

        # Use the value in the first line of the pidfile
        pidlist=`/bin/head -n1 "${pidfile}"`
        # This can optionally be written as 'sed 1q' to repalce 'head -n1'
        # should LFS move /bin/head to /usr/bin/head
    else
        # Use pidof
        pidlist=`pidof "${program}"`
    fi

    # Figure out if all listed PIDs are running.
    for pid in ${pidlist}; do
        kill -0 ${pid} 2> /dev/null

        if [ "${?}" -eq "0" ]; then
            lpids="${lpids}${pid} "
        else
            exitstatus="1"
        fi
    done

    if [ -z "${lpids}" -a ! -f "${pidfile}" ]; then
        return 3
    else
        echo "${lpids}"
        return "${exitstatus}"
```

```
      fi
}


################################################################################
# statusproc()                                                                 #
# Usage: statusproc [-p pidfile] pathname                                      #
#                                                                              #
# Purpose: This function prints the status of a particular daemon to stdout    #
#                                                                              #
# Inputs: -p pidfile, use the specified pidfile instead of pidof               #
#         pathname, path to the specified program                             #
#                                                                              #
# Return values:                                                               #
#       0 - Status printed                                                     #
#       1 - Input error. The daemon to check was not specified.                #
################################################################################
statusproc()
{
   local pidfile
   local pidlist

   if [ "${#}" = "0" ]; then
      echo "Usage: statusproc [-p pidfle] {program}"
      exit 1
   fi

   # Process arguments
   while true; do
       case "${1}" in

           -p)
               pidfile="${2}"
               shift 2
               ;;

           *)
               if [ -n "${2}" ]; then
                   echo "Too many arguments"
                   return 1
               else
                   break
               fi
               ;;
       esac
   done

   if [ -n "${pidfile}" ]; then
      pidlist=`pidofproc -p "${pidfile}" $@`
   else
      pidlist=`pidofproc $@`
   fi

   # Trim trailing blanks
   pidlist=`echo "${pidlist}" | sed -r 's/ +$//'`

   base="${1##*/}"
```

```
      if [ -n "${pidlist}" ]; then
         /bin/echo -e "${INFO}${base} is running with Process" \
            "ID(s) ${pidlist}.${NORMAL}"
      else
         if [ -n "${base}" -a -e "/var/run/${base}.pid" ]; then
            /bin/echo -e "${WARNING}${1} is not running but" \
               "/var/run/${base}.pid exists.${NORMAL}"
         else
            if [ -n "${pidfile}" -a -e "${pidfile}" ]; then
               /bin/echo -e "${WARNING}${1} is not running" \
                  "but ${pidfile} exists.${NORMAL}"
            else
               /bin/echo -e "${INFO}${1} is not running.${NORMAL}"
            fi
         fi
      fi
}

###############################################################################
# timespec()                                                                  #
#                                                                             #
# Purpose: An internal utility function to format a timestamp                 #
#          a boot log file.  Sets the STAMP variable.                         #
#                                                                             #
# Return value: Not used                                                      #
###############################################################################
timespec()
{
   STAMP="$(echo `date +"%b %d %T %:z"` `hostname`) "
   return 0
}

###############################################################################
# log_success_msg()                                                           #
# Usage: log_success_msg ["message"]                                          #
#                                                                             #
# Purpose: Print a successful status message to the screen and                #
#          a boot log file.                                                   #
#                                                                             #
# Inputs: $@ - Message                                                        #
#                                                                             #
# Return values: Not used                                                     #
###############################################################################
log_success_msg()
{
    /bin/echo -n -e "${BMPREFIX}${@}"
    /bin/echo -e "${CURS_ZERO}${SUCCESS_PREFIX}${SET_COL}${SUCCESS_SUFFIX}"

    # Strip non-printable characters from log file
    logmessage=`echo "${@}" | sed 's/\\\033[^a-zA-Z]*.//g'`

    timespec
    /bin/echo -e "${STAMP} ${logmessage} OK" >> ${BOOTLOG}

    return 0
}
```

```
log_success_msg2()
{
    /bin/echo -n -e "${BMPREFIX}${@}"
    /bin/echo -e "${CURS_ZERO}${SUCCESS_PREFIX}${SET_COL}${SUCCESS_SUFFIX}"

    echo " OK" >> ${BOOTLOG}

    return 0
}


###############################################################################
# log_failure_msg()                                                           #
# Usage: log_failure_msg ["message"]                                          #
#                                                                             #
# Purpose: Print a failure status message to the screen and                   #
#          a boot log file.                                                   #
#                                                                             #
# Inputs: $@ - Message                                                        #
#                                                                             #
# Return values: Not used                                                     #
###############################################################################
log_failure_msg()
{
    /bin/echo -n -e "${BMPREFIX}${@}"
    /bin/echo -e "${CURS_ZERO}${FAILURE_PREFIX}${SET_COL}${FAILURE_SUFFIX}"

    # Strip non-printable characters from log file

    timespec
    logmessage=`echo "${@}" | sed 's/\\\033[^a-zA-Z]*.//g'`
    /bin/echo -e "${STAMP} ${logmessage} FAIL" >> ${BOOTLOG}

    return 0
}

log_failure_msg2()
{
    /bin/echo -n -e "${BMPREFIX}${@}"
    /bin/echo -e "${CURS_ZERO}${FAILURE_PREFIX}${SET_COL}${FAILURE_SUFFIX}"

    echo "FAIL" >> ${BOOTLOG}

    return 0
}


###############################################################################
# log_warning_msg()                                                           #
# Usage: log_warning_msg ["message"]                                          #
#                                                                             #
# Purpose: Print a warning status message to the screen and                   #
#          a boot log file.                                                   #
#                                                                             #
# Return values: Not used                                                     #
###############################################################################
log_warning_msg()
{
    /bin/echo -n -e "${BMPREFIX}${@}"
```

```
    /bin/echo -e "${CURS_ZERO}${WARNING_PREFIX}${SET_COL}${WARNING_SUFFIX}"

    # Strip non-printable characters from log file
    logmessage=`echo "${@}" | sed 's/\\\033[^a-zA-Z]*.//g'`
    timespec
    /bin/echo -e "${STAMP} ${logmessage} WARN" >> ${BOOTLOG}

    return 0
}


################################################################################
# log_info_msg()                                                               #
# Usage: log_info_msg message                                                  #
#                                                                              #
# Purpose: Print an information message to the screen and                      #
#          a boot log file.  Does not print a trailing newline character.      #
#                                                                              #
# Return values: Not used                                                      #
################################################################################
log_info_msg()
{
    /bin/echo -n -e "${BMPREFIX}${@}"

    # Strip non-printable characters from log file
    logmessage=`echo "${@}" | sed 's/\\\033[^a-zA-Z]*.//g'`
    timespec
    /bin/echo -n -e "${STAMP} ${logmessage}" >> ${BOOTLOG}

    return 0
}

log_info_msg2()
{
    /bin/echo -n -e "${@}"

    # Strip non-printable characters from log file
    logmessage=`echo "${@}" | sed 's/\\\033[^a-zA-Z]*.//g'`
    /bin/echo -n -e "${logmessage}" >> ${BOOTLOG}

    return 0
}

################################################################################
# evaluate_retval()                                                            #
# Usage: Evaluate a return value and print success or failyure as appropriate  #
#                                                                              #
# Purpose: Convenience function to terminate an info message                   #
#                                                                              #
# Return values: Not used                                                      #
################################################################################
evaluate_retval()
{
    local error_value="${?}"

    if [ ${error_value} = 0 ]; then
      log_success_msg2
    else
```

```
        log_failure_msg2
    fi
}


##############################################################################
# check_signal()                                                             #
# Usage: check_signal [ -{signal} | {signal} ]                               #
#                                                                            #
# Purpose: Check for a valid signal.  This is not defined by any LSB draft,  #
#          however, it is required to check the signals to determine if the  #
#          signals chosen are invalid arguments to the other functions.      #
#                                                                            #
# Inputs: Accepts a single string value in the form or -{signal} or {signal} #
#                                                                            #
# Return values:                                                             #
#       0 - Success (signal is valid                                         #
#       1 - Signal is not valid                                              #
##############################################################################
check_signal()
{
    local valsig

    # Add error handling for invalid signals
    valsig="-ALRM -HUP -INT -KILL -PIPE -POLL -PROF -TERM -USR1 -USR2"
    valsig="${valsig} -VTALRM -STKFLT -PWR -WINCH -CHLD -URG -TSTP -TTIN"
    valsig="${valsig} -TTOU -STOP -CONT -ABRT -FPE -ILL -QUIT -SEGV -TRAP"
    valsig="${valsig} -SYS -EMT -BUS -XCPU -XFSZ -0 -1 -2 -3 -4 -5 -6 -8 -9"
    valsig="${valsig} -11 -13 -14 -15"

    echo "${valsig}" | grep -- " ${1} " > /dev/null

    if [ "${?}" -eq "0" ]; then
        return 0
    else
        return 1
    fi
}


##############################################################################
# check_sig_type()                                                           #
# Usage: check_signal [ -{signal} | {signal} ]                               #
#                                                                            #
# Purpose: Check if signal is a program termination signal or a control signal #
#          This is not defined by any LSB draft, however, it is required to  #
#          check the signals to determine if they are intended to end a      #
#          program or simply to control it.                                  #
#                                                                            #
# Inputs: Accepts a single string value in the form or -{signal} or {signal} #
#                                                                            #
# Return values:                                                             #
#       0 - Signal is used for program termination                           #
#       1 - Signal is used for program control                               #
##############################################################################
check_sig_type()
{
    local valsig
```

```
      # The list of termination signals (limited to generally used items)
      valsig="-ALRM -INT -KILL -TERM -PWR -STOP -ABRT -QUIT -2 -3 -6 -9 -14 -15"

      echo "${valsig}" | grep -- " ${1} " > /dev/null

      if [ "${?}" -eq "0" ]; then
          return 0
      else
          return 1
      fi
}


###############################################################################
# wait_for_user()                                                             #
#                                                                             #
# Purpose: Wait for the user to respond if not a headless system             #
#                                                                             #
###############################################################################
wait_for_user()
{
    # Wait for the user by default
    [ "${HEADLESS=0}" = "0" ] && read ENTER
    return 0
}


###############################################################################
# is_true()                                                                   #
#                                                                             #
# Purpose: Utility to test if a variable is true | yes | 1                    #
#                                                                             #
###############################################################################
is_true()
{
    [ "$1" = "1" ] || [ "$1" = "yes" ] || [ "$1" = "true" ] ||  [ "$1" = "y" ] ||
    [ "$1" = "t" ]
}


# End /lib/lsb/init-functions
```

D.3. /etc/rc.d/init.d/functions

```
#!/bin/sh
########################################################################
# Begin boot functions
#
# Description : Run Level Control Functions
#
# Authors     : Gerard Beekmans - gerard AT linuxfromscratch D0T org
# Update      : Bruce Dubbs - bdubbs AT linuxfromscratch D0T org
#
# Version     : LFS 7.0
#
# Notes       : With code based on Matthias Benkmann's simpleinit-msb
#               http://winterdrache.de/linux/newboot/index.html
#
#               This file is only present for backward BLFS compatibility
```

```
#
##############################################################################

## Environmental setup
# Setup default values for environment
umask 022
export PATH="/bin:/usr/bin:/sbin:/usr/sbin"

# Signal sent to running processes to refresh their configuration
RELOADSIG="HUP"

# Number of seconds between STOPSIG and FALLBACK when stopping processes
KILLDELAY="3"

## Screen Dimensions
# Find current screen size
if [ -z "${COLUMNS}" ]; then
   COLUMNS=$(stty size)
   COLUMNS=${COLUMNS##* }
fi

# When using remote connections, such as a serial port, stty size returns 0
if [ "${COLUMNS}" = "0" ]; then
   COLUMNS=80
fi

## Measurements for positioning result messages
COL=$((${COLUMNS} - 8))
WCOL=$((${COL} - 2))

## Provide an echo that supports -e and -n
# If formatting is needed, $ECHO should be used
case "`echo -e -n test`" in
   -[en]*)
      ECHO=/bin/echo
      ;;
   *)
      ECHO=echo
      ;;
esac

## Set Cursor Position Commands, used via $ECHO
SET_COL="\\033[${COL}G"      # at the $COL char
SET_WCOL="\\033[${WCOL}G"    # at the $WCOL char
CURS_UP="\\033[1A\\033[0G"   # Up one line, at the 0'th char

## Set color commands, used via $ECHO
# Please consult `man console_codes for more information
# under the "ECMA-48 Set Graphics Rendition" section
#
# Warning: when switching from a 8bit to a 9bit font,
# the linux console will reinterpret the bold (1;) to
# the top 256 glyphs of the 9bit font.  This does
# not affect framebuffer consoles
NORMAL="\\033[0;39m"         # Standard console grey
SUCCESS="\\033[1;32m"        # Success is green
WARNING="\\033[1;33m"        # Warnings are yellow
```

```
FAILURE="\\033[1;31m"          # Failures are red
INFO="\\033[1;36m"             # Information is light cyan
BRACKET="\\033[1;34m"          # Brackets are blue

STRING_LENGTH="0"   # the length of the current message

#*******************************************************************************
# Function - boot_mesg()
#
# Purpose:      Sending information from bootup scripts to the console
#
# Inputs:       $1 is the message
#               $2 is the colorcode for the console
#
# Outputs:      Standard Output
#
# Dependencies: - sed for parsing strings.
#          - grep for counting string length.
#
# Todo:
#*******************************************************************************
boot_mesg()
{
   local ECHOPARM=""

   while true
   do
      case "${1}" in
         -n)
            ECHOPARM=" -n "
            shift 1
            ;;
         -*)
            echo "Unknown Option: ${1}"
            return 1
            ;;
         *)
            break
            ;;
      esac
   done

   ## Figure out the length of what is to be printed to be used
   ## for warning messages.
   STRING_LENGTH=$((${#1} + 1))

   # Print the message to the screen
   ${ECHO} ${ECHOPARM} -e "${2}${1}"

   # Log the message
   [ -d /run/var ] || return
   ${ECHO} ${ECHOPARM} -e "${2}${1}" >> /run/var/bootlog
}

boot_mesg_flush()
{
   # Reset STRING_LENGTH for next message
```

```
    STRING_LENGTH="0"
}

echo_ok()
{
    ${ECHO} -n -e "${CURS_UP}${SET_COL}${BRACKET}[${SUCCESS}  OK  ${BRACKET}]"
    ${ECHO} -e "${NORMAL}"
    boot_mesg_flush

    [ -d /run/var ] || return
    ${ECHO} -e "[ OK ]" >> /run/var/bootlog
}

echo_failure()
{
    ${ECHO} -n -e "${CURS_UP}${SET_COL}${BRACKET}[${FAILURE} FAIL ${BRACKET}]"
    ${ECHO} -e "${NORMAL}"
     boot_mesg_flush

    [ -d /run/var ] || return
    ${ECHO} -e "[ FAIL]"  >> /run/var/bootlog
}

echo_warning()
{
    ${ECHO} -n -e "${CURS_UP}${SET_COL}${BRACKET}[${WARNING} WARN ${BRACKET}]"
    ${ECHO} -e "${NORMAL}"
    boot_mesg_flush

    [ -d /run/var ] || return
    ${ECHO} -e "[ WARN ]"  >> /run/var/bootlog
}

echo_skipped()
{
    ${ECHO} -n -e "${CURS_UP}${SET_COL}${BRACKET}[${WARNING} SKIP ${BRACKET}]"
    ${ECHO} -e "${NORMAL}"
    boot_mesg_flush

    [ -d /run/var ] || return
    ${ECHO} -e "  [ SKIP ]" >> /run/var/bootlog
}

wait_for_user()
{
    # Wait for the user by default
    [ "${HEADLESS=0}" = "0" ] && read ENTER
}

evaluate_retval()
{
    error_value="${?}"

    if [ ${error_value} = 0 ]; then
       echo_ok
    else
       echo_failure
```

```
    fi

    # This prevents the 'An Unexpected Error Has Occurred' from trivial
    # errors.
    return 0
}

print_status()
{
    if [ "${#}" = "0" ]; then
        echo "Usage: ${0} {success|warning|failure}"
        return 1
    fi

    case "${1}" in

        success)
            echo_ok
            ;;

        warning)
            # Leave this extra case in because old scripts
            # may call it this way.
            case "${2}" in
                running)
                    ${ECHO} -e -n "${CURS_UP}"
                    ${ECHO} -e -n "\\033[${STRING_LENGTH}G   "
                    boot_mesg "Already running." ${WARNING}
                    echo_warning
                    ;;
                not_running)
                    ${ECHO} -e -n "${CURS_UP}"
                    ${ECHO} -e -n "\\033[${STRING_LENGTH}G   "
                    boot_mesg "Not running." ${WARNING}
                    echo_warning
                    ;;
                not_available)
                    ${ECHO} -e -n "${CURS_UP}"
                    ${ECHO} -e -n "\\033[${STRING_LENGTH}G   "
                    boot_mesg "Not available." ${WARNING}
                    echo_warning
                    ;;
                *)
                    # This is how it is supposed to
                    # be called
                    echo_warning
                    ;;
            esac
        ;;

        failure)
            echo_failure
        ;;

    esac

}
```

```
reloadproc()
{
    local pidfile=""
    local failure=0

    while true
    do
        case "${1}" in
            -p)
                pidfile="${2}"
                shift 2
                ;;
            -*)
                log_failure_msg "Unknown Option: ${1}"
                return 2
                ;;
            *)
                break
                ;;
        esac
    done

    if [ "${#}" -lt "1" ]; then
        log_failure_msg "Usage: reloadproc [-p pidfile] pathname"
        return 2
    fi

    # This will ensure compatibility with previous LFS Bootscripts
    if [ -n "${PIDFILE}" ]; then
        pidfile="${PIDFILE}"
    fi

    # Is the process running?
    if [ -z "${pidfile}" ]; then
        pidofproc -s "${1}"
    else
        pidofproc -s -p "${pidfile}" "${1}"
    fi

    # Warn about stale pid file
    if [ "$?" = 1 ]; then
        boot_mesg -n "Removing stale pid file: ${pidfile}. " ${WARNING}
        rm -f "${pidfile}"
    fi

    if [ -n "${pidlist}" ]; then
        for pid in ${pidlist}
        do
            kill -"${RELOADSIG}" "${pid}" || failure="1"
        done

        (exit ${failure})
        evaluate_retval

    else
        boot_mesg "Process ${1} not running." ${WARNING}
```

```
            echo_warning
    fi
}

statusproc()
{
    local pidfile=""
    local base=""
    local ret=""

    while true
    do
        case "${1}" in
            -p)
                pidfile="${2}"
                shift 2
                ;;
            -*)
                log_failure_msg "Unknown Option: ${1}"
                return 2
                ;;
            *)
                break
                ;;
        esac
    done

    if [ "${#}" != "1" ]; then
        shift 1
        log_failure_msg "Usage: statusproc [-p pidfile] pathname"
        return 2
    fi

    # Get the process basename
    base="${1##*/}"

    # This will ensure compatibility with previous LFS Bootscripts
    if [ -n "${PIDFILE}" ]; then
        pidfile="${PIDFILE}"
    fi

    # Is the process running?
    if [ -z "${pidfile}" ]; then
        pidofproc -s "${1}"
    else
        pidofproc -s -p "${pidfile}" "${1}"
    fi

    # Store the return status
    ret=$?

    if [ -n "${pidlist}" ]; then
        ${ECHO} -e "${INFO}${base} is running with Process"\
            "ID(s) ${pidlist}.${NORMAL}"
    else
        if [ -n "${base}" -a -e "/var/run/${base}.pid" ]; then
            ${ECHO} -e "${WARNING}${1} is not running but"\
```

```
                        "/var/run/${base}.pid exists.${NORMAL}"
            else
                if [ -n "${pidfile}" -a -e "${pidfile}" ]; then
                    ${ECHO} -e "${WARNING}${1} is not running"\
                        "but ${pidfile} exists.${NORMAL}"
                else
                    ${ECHO} -e "${INFO}${1} is not running.${NORMAL}"
                fi
            fi
        fi

    # Return the status from pidofproc
    return $ret
}

# The below functions are documented in the LSB-generic 2.1.0

#*******************************************************************************
# Function - pidofproc [-s] [-p pidfile] pathname
#
# Purpose: This function returns one or more pid(s) for a particular daemon
#
# Inputs: -p pidfile, use the specified pidfile instead of pidof
#         pathname, path to the specified program
#
# Outputs: return 0 - Success, pid's in stdout
#          return 1 - Program is dead, pidfile exists
#          return 2 - Invalid or excessive number of arguments,
#                     warning in stdout
#          return 3 - Program is not running
#
# Dependencies: pidof, echo, head
#
# Todo: Remove dependency on head
#       This replaces getpids
#       Test changes to pidof
#
#*******************************************************************************
pidofproc()
{
    local pidfile=""
    local lpids=""
    local silent=""
    pidlist=""
    while true
    do
        case "${1}" in
            -p)
                pidfile="${2}"
                shift 2
                ;;

            -s)
                # Added for legacy opperation of getpids
                # eliminates several '> /dev/null'
                silent="1"
                shift 1
```

```
                ;;
            -*)
                log_failure_msg "Unknown Option: ${1}"
                return 2
                ;;
            *)
                break
                ;;
        esac
    done

    if [ "${#}" != "1" ]; then
        shift 1
        log_failure_msg "Usage: pidofproc [-s] [-p pidfile] pathname"
        return 2
    fi

    if [ -n "${pidfile}" ]; then
        if [ ! -r "${pidfile}" ]; then
            return 3 # Program is not running
        fi

        lpids=`head -n 1 ${pidfile}`
        for pid in ${lpids}
        do
            if [ "${pid}" -ne "$$" -a "${pid}" -ne "${PPID}" ]; then
                kill -0 "${pid}" 2>/dev/null &&
                pidlist="${pidlist} ${pid}"
            fi

            if [ "${silent}" != "1" ]; then
                echo "${pidlist}"
            fi

            test -z "${pidlist}" &&
            # Program is dead, pidfile exists
            return 1
            # else
            return 0
        done

    else
        pidlist=`pidof -o $$ -o $PPID -x "$1"`
        if [ "${silent}" != "1" ]; then
            echo "${pidlist}"
        fi

        # Get provide correct running status
        if [ -n "${pidlist}" ]; then
            return 0
        else
            return 3
        fi

    fi

    if [ "$?" != "0" ]; then
```

```
        return 3 # Program is not running
   fi
}

#*******************************************************************************
# Function - loadproc [-f] [-n nicelevel] [-p pidfile] pathname [args]
#
# Purpose: This runs the specified program as a daemon
#
# Inputs: -f, run the program even if it is already running
#         -n nicelevel, specifies a nice level. See nice(1).
#         -p pidfile, uses the specified pidfile
#         pathname, pathname to the specified program
#         args, arguments to pass to specified program
#
# Outputs: return 0 - Success
#          return 2 - Invalid of excessive number of arguments,
#                     warning in stdout
#          return 4 - Program or service status is unknown
#
# Dependencies: nice, rm
#
# Todo: LSB says this should be called start_daemon
#       LSB does not say that it should call evaluate_retval
#       It checks for PIDFILE, which is deprecated.
#         Will be removed after BLFS 6.0
#       loadproc returns 0 if program is already running, not LSB compliant
#
#*******************************************************************************
loadproc()
{
   local pidfile=""
   local forcestart=""
   local nicelevel="10"

# This will ensure compatibility with previous LFS Bootscripts
   if [ -n "${PIDFILE}" ]; then
      pidfile="${PIDFILE}"
   fi

  while true
   do
      case "${1}" in
         -f)
            forcestart="1"
            shift 1
            ;;
         -n)
            nicelevel="${2}"
            shift 2
            ;;
         -p)
            pidfile="${2}"
            shift 2
            ;;
         -*)
            log_failure_msg "Unknown Option: ${1}"
```

```
                return 2 #invalid or excess argument(s)
                ;;
            *)
                break
                ;;
        esac
    done

    if [ "${#}" = "0" ]; then
        log_failure_msg "Usage: loadproc [-f] [-n nicelevel] [-p pidfile] pathname [arg
        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  ""${BRACKET}""]""${NORMAI

   [ -d /run/var ] || return 0
   ${ECHO} -n -e "${@}  [ OK ]"  >> /run/var/bootlog
   return 0
}


#********************************************************************************
# Function - log_failure_msg "message"
#
# Purpose: Print a failure message
#
# Inputs: $@ - Message
#
# Outputs: Text output to screen
#
# Dependencies: echo
#
# Todo: logging
#
#********************************************************************************
log_failure_msg() {
   ${ECHO} -n -e "${BOOTMESG_PREFIX}${@}"
   ${ECHO} -e "${SET_COL}""${BRACKET}""[""${FAILURE}"" FAIL ""${BRACKET}""]""${NORMAI

   [ -d /run/var ] || return 0
   ${ECHO} -e "${@}  [ FAIL ]" >> /run/var/bootlog
   return 0
}


#********************************************************************************
# Function - log_warning_msg "message"
#
# Purpose: print a warning message
#
# Inputs: $@ - Message
#
# Outputs: Text output to screen
#
# Dependencies: echo
```

```
#
# Todo: logging
#
#******************************************************************************
log_warning_msg() {
    ${ECHO} -n -e "${BOOTMESG_PREFIX}${@}"
    ${ECHO} -e "${SET_COL}""${BRACKET}""[""${WARNING}"" WARN ""${BRACKET}""]""${NORMAL

    [ -d /run/var ] || return 0
    ${ECHO} -e "${@}  [ WARN ]" >> /run/var/bootlog
    return 0
}


#******************************************************************************
# Function - log_skipped_msg "message"
#
# Purpose: print a message that the script was skipped
#
# Inputs: $@ - Message
#
# Outputs: Text output to screen
#
# Dependencies: echo
#
# Todo: logging
#
#******************************************************************************
log_skipped_msg() {
    ${ECHO} -n -e "${BOOTMESG_PREFIX}${@}"
    ${ECHO} -e "${SET_COL}""${BRACKET}""[""${WARNING}"" SKIP ""${BRACKET}""]""${NORMAL

    [ -d /run/var ] || return 0
    ${ECHO} -e "${@}  [ SKIP ]" >> /run/var/bootlog
    return 0
}


# End boot functions
```

D.4. /etc/rc.d/init.d/mountvirtfs

```
#!/bin/sh
########################################################################
# Begin mountvirtfs
#
# Description : Mount proc, sysfs, and run
#
# Authors     : Gerard Beekmans - gerard AT linuxfromscratch D0T org
#               DJ Lucas - dj AT linuxfromscratch D0T org
# Update      : Bruce Dubbs - bdubbs AT linuxfromscratch D0T org
#
# Version     : LFS 7.0
#
########################################################################

### BEGIN INIT INFO
# Provides:            mountvirtfs
```

```
# Required-Start:
# Should-Start:
# Required-Stop:
# Should-Stop:
# Default-Start:        S
# Default-Stop:
# Short-Description:    Mounts /sys and /proc virtual (kernel) filesystems.
#                       Mounts /run (tmpfs) and /dev (devtmpfs).
# Description:          Mounts /sys and /proc virtual (kernel) filesystems.
#                       Mounts /run (tmpfs) and /dev (devtmpfs).
# X-LFS-Provided-By:    LFS
### END INIT INFO

. /lib/lsb/init-functions

case "${1}" in
   start)
      # Make sure /run/var is available before logging any messages
      if ! mountpoint /run >/dev/null; then
         mount -n /run || failed=1
      fi

      mkdir -p /run/var /run/lock /run/shm
      chmod 1777 /run/shm

      log_info_msg "Mounting virtual file systems: ${INFO}/run"

      if ! mountpoint /proc >/dev/null; then
         log_info_msg2 " ${INFO}/proc"
         mount -n -o nosuid,noexec,nodev /proc || failed=1
      fi

      if ! mountpoint /sys >/dev/null; then
         log_info_msg2 " ${INFO}/sys"
         mount -n -o nosuid,noexec,nodev /sys || failed=1
      fi

      if ! mountpoint /dev >/dev/null; then
         log_info_msg2 " ${INFO}/dev"
         mount -n -o mode=0755,nosuid /dev  || failed=1
      fi

      # Copy devices that Udev >= 155 doesn't handle to /dev
      cp -a /lib/udev/devices/* /dev

      ln -sfn /run/shm /dev/shm

      (exit ${failed})
      evaluate_retval
      exit $failed
      ;;

   *)
      echo "Usage: ${0} {start}"
      exit 1
      ;;
esac
```

```
# End mountvirtfs
```

D.5. /etc/rc.d/init.d/modules

```
#!/bin/sh
########################################################################
# Begin modules
#
# Description : Module auto-loading script
#
# Authors     : Zack Winkles
#               DJ Lucas - dj AT linuxfromscratch D0T org
# Update      : Bruce Dubbs - bdubbs AT linuxfromscratch D0T org
#
# Version     : LFS 7.0
#
########################################################################

### BEGIN INIT INFO
# Provides:            modules
# Required-Start:      mountvirtfs sysctl
# Should-Start:
# Required-Stop:
# Should-Stop:
# Default-Start:       S
# Default-Stop:
# Short-Description:   Loads required modules.
# Description:         Loads modules listed in /etc/sysconfig/modules.
# X-LFS-Provided-By:   LFS
### END INIT INFO

# Assure that the kernel has module support.
[ -e /proc/ksyms -o -e /proc/modules ] || exit 0

. /lib/lsb/init-functions

case "${1}" in
   start)
      # Exit if there's no modules file or there are no
      # valid entries
      [ -r /etc/sysconfig/modules ]                || exit 0
      egrep -qv '^($|#)' /etc/sysconfig/modules || exit 0

      log_info_msg "Loading modules:"

      # Only try to load modules if the user has actually given us
      # some modules to load.

      while read module args; do

         # Ignore comments and blank lines.
         case "$module" in
            ""|"#"*) continue ;;
         esac
```

```
                # Attempt to load the module, passing any arguments provided.
                modprobe ${module} ${args} >/dev/null

                # Print the module name if successful, otherwise take note.
                if [ $? -eq 0 ]; then
                    log_info_msg2 " ${module}"
                else
                    failedmod="${failedmod} ${module}"
                fi
        done < /etc/sysconfig/modules

        # Print a message about successfully loaded modules on the correct line.
        log_success_msg2

        # Print a failure message with a list of any modules that
        # may have failed to load.
        if [ -n "${failedmod}" ]; then
            log_failure_msg "Failed to load modules:${failedmod}"
            exit 1
        fi
        ;;

    *)
        echo "Usage: ${0} {start}"
        exit 1
        ;;
esac

exit 0

# End modules
```

D.6. /etc/rc.d/init.d/udev

```
#!/bin/sh
########################################################################
# Begin udev
#
# Description : Udev cold-plugging script
#
# Authors     : Zack Winkles, Alexander E. Patrakov
#               DJ Lucas - dj AT linuxfromscratch D0T org
# Update      : Bruce Dubbs - bdubbs AT linuxfromscratch D0T org
#
# Version     : LFS 7.0
#
########################################################################

### BEGIN INIT INFO
# Provides:            udev $time
# Required-Start:
# Should-Start:        modules
# Required-Stop:
# Should-Stop:
# Default-Start:       S
# Default-Stop:
```

```
# Short-Description:    Populates /dev with device nodes.
# Description:          Mounts a tempfs on /dev and starts the udevd daemon.
#                       Device nodes are created as defined by udev.
# X-LFS-Provided-By:    LFS
### END INIT INFO

. /lib/lsb/init-functions

case "${1}" in
    start)
        log_info_msg "Populating /dev with device nodes... "
        if ! grep -q '[[:space:]]sysfs' /proc/mounts; then
            log_failure_msg2
            msg="FAILURE:\n\nUnable to create "
            msg="${msg}devices without a SysFS filesystem\n\n"
            msg="${msg}After you press Enter, this system "
            msg="${msg}will be halted and powered off.\n\n"
            log_info_msg "$msg"
            log_info_msg "Press Enter to continue..."
            wait_for_user
            /etc/rc.d/init.d/halt stop
        fi

        # Udev handles uevents itself, so we don't need to have
        # the kernel call out to any binary in response to them
        echo > /proc/sys/kernel/hotplug

        # Start the udev daemon to continually watch for, and act on,
        # uevents
        /lib/udev/udevd --daemon

        # Now traverse /sys in order to "coldplug" devices that have
        # already been discovered
        /sbin/udevadm trigger --action=add    --type=subsystems
        /sbin/udevadm trigger --action=add    --type=devices
        /sbin/udevadm trigger --action=change --type=devices

        # Now wait for udevd to process the uevents we triggered
        if ! is_true "$OMIT_UDEV_SETTLE"; then
            /sbin/udevadm settle
        fi

        # If any LVM based partitions are on the system, ensure they
        # are activated so they can be used.
        if [ -x /sbin/vgchange ]; then /sbin/vgchange -a y >/dev/null; fi

        log_success_msg2
        ;;

    *)
        echo "Usage ${0} {start}"
        exit 1
        ;;
esac

exit 0
```

```
# End udev
```

D.7. /etc/rc.d/init.d/swap

```
#!/bin/sh
########################################################################
# Begin swap
#
# Description : Swap Control Script
#
# Authors     : Gerard Beekmans - gerard AT linuxfromscratch D0T org
#               DJ Lucas - dj AT linuxfromscratch D0T org
# Update      : Bruce Dubbs - bdubbs AT linuxfromscratch D0T org
#
# Version     : LFS 7.0
#
########################################################################

### BEGIN INIT INFO
# Provides:           swap
# Required-Start:     udev
# Should-Start:       modules
# Required-Stop:      localnet
# Should-Stop:
# Default-Start:      S
# Default-Stop:       0 6
# Short-Description:  Mounts and unmounts swap partitions.
# Description:        Mounts and unmounts swap partitions defined in
#                     /etc/fstab.
# X-LFS-Provided-By:  LFS
### END INIT INFO

. /lib/lsb/init-functions

case "${1}" in
   start)
      log_info_msg "Activating all swap files/partitions..."
      swapon -a
      evaluate_retval
      ;;

   stop)
      log_info_msg "Deactivating all swap files/partitions..."
      swapoff -a
      evaluate_retval
      ;;

   restart)
      ${0} stop
      sleep 1
      ${0} start
      ;;

   status)
      log_success_msg "Retrieving swap status."
      swapon -s
```

```
        ;;

    *)
        echo "Usage: ${0} {start|stop|restart|status}"
        exit 1
        ;;
esac

exit 0

# End swap
```

D.8. /etc/rc.d/init.d/setclock

```
#!/bin/sh
########################################################################
# Begin setclock
#
# Description : Setting Linux Clock
#
# Authors     : Gerard Beekmans - gerard AT linuxfromscratch D0T org
#               DJ Lucas - dj AT linuxfromscratch D0T org
# Update      : Bruce Dubbs - bdubbs AT linuxfromscratch D0T org
#
# Version     : LFS 7.0
#
########################################################################

### BEGIN INIT INFO
# Provides:
# Required-Start:
# Should-Start:        modules
# Required-Stop:
# Should-Stop:         $syslog
# Default-Start:       S
# Default-Stop:
# Short-Description:   Stores and restores time from the hardware clock
# Description:         On boot, system time is obtained from hwclock.  The
#                      hardware clock can also be set on shutdown.
# X-LFS-Provided-By:   LFS BLFS
### END INIT INFO

. /lib/lsb/init-functions

[ -r /etc/sysconfig/clock ] && . /etc/sysconfig/clock


case "${UTC}" in
    yes|true|1)
        CLOCKPARAMS="${CLOCKPARAMS} --utc"
        ;;

    no|false|0)
        CLOCKPARAMS="${CLOCKPARAMS} --localtime"
        ;;
```

```
esac

case ${1} in
   start)
      hwclock --hctosys ${CLOCKPARAMS} >/dev/null
      ;;

   stop)
      log_info_msg "Setting hardware clock..."
      hwclock --systohc ${CLOCKPARAMS} >/dev/null
      evaluate_retval
      ;;

   *)
      echo "Usage: ${0} {start|stop}"
      exit 1
      ;;

esac

exit 0
```

D.9. /etc/rc.d/init.d/checkfs

```
#!/bin/sh
########################################################################
# Begin checkfs
#
# Description : File System Check
#
# Authors     : Gerard Beekmans - gerard AT linuxfromscratch D0T org
#               A. Luebke - luebke@users.sourceforge.net
#               DJ Lucas - dj AT linuxfromscratch D0T org
# Update      : Bruce Dubbs - bdubbs AT linuxfromscratch D0T org
#
# Version     : LFS 7.0
#
# Based on checkfs script from LFS-3.1 and earlier.
#
# From man fsck
# 0    - No errors
# 1    - File system errors corrected
# 2    - System should be rebooted
# 4    - File system errors left uncorrected
# 8    - Operational error
# 16   - Usage or syntax error
# 32   - Fsck canceled by user request
# 128  - Shared library error
#
########################################################################

### BEGIN INIT INFO
# Provides:            checkfs
# Required-Start:      udev swap $time
# Should-Start:
# Required-Stop:
```

```
# Should-Stop:
# Default-Start:        S
# Default-Stop:
# Short-Description:    Checks local filesystems before mounting.
# Description:          Checks local filesystmes before mounting.
# X-LFS-Provided-By:    LFS
### END INIT INFO

. /lib/lsb/init-functions

case "${1}" in
   start)
      if [ -f /fastboot ]; then
         msg="/fastboot found, will omit "
         msg="${msg} file system checks as requested.\n"
         log_info_msg "${msg}"
         exit 0
      fi

      log_info_msg "Mounting root file system in read-only mode... "
      mount -n -o remount,ro / >/dev/null

      if [ ${?} != 0 ]; then
         log_failure_msg2
         msg="\n\nCannot check root "
         msg="${msg}filesystem because it could not be mounted "
         msg="${msg}in read-only mode.\n\n"
         msg="${msg}After you press Enter, this system will be "
         msg="${msg}halted and powered off.\n\n"
         log_failure_msg "${msg}"

         log_info_msg "Press Enter to continue..."
         wait_for_user
         /etc/rc.d/init.d/halt stop
      else
         log_success_msg2
      fi

      if [ -f /forcefsck ]; then
         msg="\n/forcefsck found, forcing file"
         msg="${msg} system checks as requested."
         log_success_msg "$msg"
         options="-f"
      else
         options=""
      fi

      log_info_msg "Checking file systems..."
      # Note: -a option used to be -p; but this fails e.g. on fsck.minix
      if is_true "$VERBOSE_FSCK"; then
        fsck ${options} -a -A -C -T
      else
        fsck ${options} -a -A -C -T >/dev/null
      fi

      error_value=${?}
```

```
        if [ "${error_value}" = 0 ]; then
            log_success_msg2
        fi

        if [ "${error_value}" = 1 ]; then
            msg="\nWARNING:\n\nFile system errors "
            msg="${msg}were found and have been corrected.\n"
            msg="${msg}You may want to double-check that "
            msg="${msg}everything was fixed properly."
            log_warning_msg "$msg"
        fi

        if [ "${error_value}" = 2 -o "${error_value}" = 3 ]; then
            msg="\nWARNING:\n\nFile system errors "
            msg="${msg}were found and have been been "
            msg="${msg}corrected, but the nature of the "
            msg="${msg}errors require this system to be rebooted.\n\n"
            msg="${msg}After you press enter, "
            msg="${msg}this system will be rebooted\n\n"
            log_failure_msg "$msg"

            log_info_msg "Press Enter to continue..."
            wait_for_user
            reboot -f
        fi

        if [ "${error_value}" -gt 3 -a "${error_value}" -lt 16 ]; then
            msg="\nFAILURE:\n\nFile system errors "
            msg="${msg}were encountered that could not be "
            msg="${msg}fixed automatically.  This system "
            msg="${msg}cannot continue to boot and will "
            msg="${msg}therefore be halted until those "
            msg="${msg}errors are fixed manually by a "
            msg="${msg}System Administrator.\n\n"
            msg="${msg}After you press Enter, this system will be "
            msg="${msg}halted and powered off.\n\n"
            log_failure_msg "$msg"

            log_info_msg "Press Enter to continue..."
            wait_for_user
            /etc/rc.d/init.d/halt stop
        fi

        if [ "${error_value}" -ge 16 ]; then
            msg="\nFAILURE:\n\nUnexpected Failure "
            msg="${msg}running fsck.  Exited with error "
            msg="${msg} code: ${error_value}."
            log_failure_msg $msg
            exit ${error_value}
        fi

        exit 0
        ;;
    *)
        echo "Usage: ${0} {start}"
        exit 1
        ;;
```

```
    esac


    # End checkfs
```

D.10. /etc/rc.d/init.d/mountfs

```
#!/bin/sh
########################################################################
# Begin mountfs
#
# Description : File System Mount Script
#
# Authors     : Gerard Beekmans - gerard AT linuxfromscratch D0T org
#               DJ Lucas - dj AT linuxfromscratch D0T org
# Update      : Bruce Dubbs - bdubbs AT linuxfromscratch D0T org
#
# Version     : LFS 7.0
#
########################################################################

### BEGIN INIT INFO
# Provides:            $local_fs
# Required-Start:      udev checkfs
# Should-Start:
# Required-Stop:       swap
# Should-Stop:
# Default-Start:       S
# Default-Stop:        0 6
# Short-Description:   Mounts/unmounts local filesystems defined in /etc/fstab.
# Description:         Remounts root filesystem read/write and mounts all
#                      remaining local filesystems defined in /etc/fstab on
#                      start.  Remounts root filesystem read-only and unmounts
#                      remaining filesystems on stop.
# X-LFS-Provided-By:   LFS
### END INIT INFO

. /lib/lsb/init-functions

case "${1}" in
   start)
      log_info_msg "Remounting root file system in read-write mode..."
      mount -n -o remount,rw / >/dev/null
      evaluate_retval

      # Remove fsck-related file system watermarks.
      rm -f /fastboot /forcefsck

      log_info_msg "Recording existing mounts in /etc/mtab..."
      > /etc/mtab

      mount -f /     || failed=1
      mount -f /proc || failed=1
      mount -f /sys  || failed=1
      mount -f /run  || failed=1
      mount -f /dev  || failed=1
      (exit ${failed})
```

```
        evaluate_retval

        # This will mount all filesystems that do not have _netdev in
        # their option list.  _netdev denotes a network filesystem.

        log_info_msg "Mounting remaining file systems..."
        mount -a -O no_netdev >/dev/null
        evaluate_retval
        exit $failed
        ;;

    stop)
        # Don't unmount tmpfs like /run
        log_info_msg "Unmounting all other currently mounted file systems..."
        umount -a -d -r -t notmpfs,nosysfs,nodevtmpfs,noproc >/dev/null
        evaluate_retval

        # Make all LVM volume groups unavailable, if appropriate
        # This fails if swap or / are on an LVM partition
        #if [ -x /sbin/vgchange ]; then /sbin/vgchange -an > /dev/null; fi
        ;;

    *)
        echo "Usage: ${0} {start|stop}"
        exit 1
        ;;
esac

# End mountfs
```

D.11. /etc/rc.d/init.d/udev_retry

```
#!/bin/sh
########################################################################
# Begin udev_retry
#
# Description : Udev cold-plugging script (retry)
#
# Authors     : Alexander E. Patrakov
#               DJ Lucas - dj AT linuxfromscratch D0T org
# Update      : Bruce Dubbs - bdubbs AT linuxfromscratch D0T org
#               Bryan Kadzban -
#
# Version     : LFS 7.0
#
########################################################################

### BEGIN INIT INFO
# Provides:            udev_retry
# Required-Start:      udev
# Should-Start:        $local_fs
# Required-Stop:
# Should-Stop:
# Default-Start:       S
# Default-Stop:
# Short-Description:   Replays failed uevents and creates additional devices.
```

```
# Description:         Replays any failed uevents that were skipped due to
#                      slow hardware initialization, and creates those needed
#                      device nodes
# X-LFS-Provided-By:   LFS
### END INIT INFO

. /lib/lsb/init-functions

case "${1}" in
   start)
      log_info_msg "Retrying failed uevents, if any..."

      # As of udev-186, the --run option is no longer valid
      #rundir=$(/sbin/udevadm info --run)
      rundir=/run/udev
      # From Debian: "copy the rules generated before / was mounted
      # read-write":

      for file in ${rundir}/tmp-rules--*; do
         dest=${file##*tmp-rules--}
         [ "$dest" = '*' ] && break
         cat $file >> /etc/udev/rules.d/$dest
         rm -f $file
      done

      # Re-trigger the uevents that may have failed,
      # in hope they will succeed now
      /bin/sed -e 's/#.*$//' /etc/sysconfig/udev_retry | /bin/grep -v '^$' | \
      while read line ; do
         for subsystem in $line ; do
             /sbin/udevadm trigger --subsystem-match=$subsystem --action=add
         done
      done

      # Now wait for udevd to process the uevents we triggered
      if ! is_true "$OMIT_UDEV_RETRY_SETTLE"; then
         /sbin/udevadm settle
      fi

      evaluate_retval
      ;;

   *)
      echo "Usage ${0} {start}"
      exit 1
      ;;
esac

exit 0

# End udev_retry
```

D.12. /etc/rc.d/init.d/cleanfs

```
#!/bin/sh
########################################################################
```

```
# Begin cleanfs
#
# Description : Clean file system
#
# Authors     : Gerard Beekmans - gerard AT linuxfromscratch DOT org
#               DJ Lucas - dj AT linuxfromscratch DOT org
# Update      : Bruce Dubbs - bdubbs AT linuxfromscratch DOT org
#
# Version     : LFS 7.0
#
########################################################################

### BEGIN INIT INFO
# Provides:          cleanfs
# Required-Start:    $local_fs
# Should-Start:
# Required-Stop:
# Should-Stop:
# Default-Start:     S
# Default-Stop:
# Short-Description: Cleans temporary directories early in the boot process.
# Description:       Cleans temporary directories /var/run, /var/lock, and
#                    optionally, /tmp.  cleanfs also creates /var/run/utmp
#                    and any files defined in /etc/sysconfig/createfiles.
# X-LFS-Provided-By: LFS
### END INIT INFO

. /lib/lsb/init-functions

# Function to create files/directory on boot.
create_files()
{
   # Input to file descriptor 9 and output to stdin (redirection)
   exec 9>&0 < /etc/sysconfig/createfiles

   while read name type perm usr grp dtype maj min junk
   do
      # Ignore comments and blank lines.
      case "${name}" in
         ""|\#*) continue ;;
      esac

      # Ignore existing files.
      if [ ! -e "${name}" ]; then
         # Create stuff based on its type.
         case "${type}" in
            dir)
               mkdir "${name}"
               ;;
            file)
               :> "${name}"
               ;;
            dev)
               case "${dtype}" in
                  char)
                     mknod "${name}" c ${maj} ${min}
                     ;;
```

```
                        block)
                            mknod "${name}" b ${maj} ${min}
                            ;;
                        pipe)
                            mknod "${name}" p
                            ;;
                        *)
                            log_warning_msg "\nUnknown device type: ${dtype}"
                            ;;
                    esac
                    ;;
                *)
                    log_warning_msg "\nUnknown type: ${type}"
                    continue
                    ;;
            esac

            # Set up the permissions, too.
            chown ${usr}:${grp} "${name}"
            chmod ${perm} "${name}"
        fi
    done

    # Close file descriptor 9 (end redirection)
    exec 0>&9 9>&-
    return 0
}

case "${1}" in
    start)
        log_info_msg "Cleaning file systems:"

        if [ "${SKIPTMPCLEAN}" = "" ]; then
            log_info_msg2 " /tmp"
            cd /tmp &&
            find . -xdev -mindepth 1 ! -name lost+found -delete || failed=1
        fi

        > /var/run/utmp

        if grep -q '^utmp:' /etc/group ; then
            chmod 664 /var/run/utmp
            chgrp utmp /var/run/utmp
        fi

        (exit ${failed})
        evaluate_retval

        if egrep -qv '^(#|$)' /etc/sysconfig/createfiles 2>/dev/null; then
            log_info_msg "Creating files and directories... "
            create_files     # Always returns 0
            evaluate_retval
        fi

        exit $failed
        ;;
    *)
```

```
        echo "Usage: ${0} {start}"
        exit 1
        ;;
esac

# End cleanfs
```

D.13. /etc/rc.d/init.d/console

```
#!/bin/sh
########################################################################
# Begin console
#
# Description : Sets keymap and screen font
#
# Authors     : Gerard Beekmans - gerard AT linuxfromscratch D0T org
#               Alexander E. Patrakov
#               DJ Lucas - dj AT linuxfromscratch D0T org
# Update      : Bruce Dubbs - bdubbs AT linuxfromscratch D0T org
#
# Version     : LFS 7.0
#
########################################################################

### BEGIN INIT INFO
# Provides:          console
# Required-Start:
# Should-Start:      $local_fs
# Required-Stop:
# Should-Stop:
# Default-Start:     S
# Default-Stop:
# Short-Description: Sets up a localised console.
# Description:       Sets up fonts and language settings for the user's
#                    local as defined by /etc/sysconfig/console.
# X-LFS-Provided-By: LFS
### END INIT INFO

. /lib/lsb/init-functions

# Native English speakers probably don't have /etc/sysconfig/console at all
[ -r /etc/sysconfig/console ] && . /etc/sysconfig/console

is_true()
{
   [ "$1" = "1" ] || [ "$1" = "yes" ] || [ "$1" = "true" ]
}

failed=0

case "${1}" in
   start)
      # See if we need to do anything
      if [ -z "${KEYMAP}"          ] && [ -z "${KEYMAP_CORRECTIONS}" ] &&
         [ -z "${FONT}"            ] && [ -z "${LEGACY_CHARSET}"     ] &&
         ! is_true "${UNICODE}"; then
```

```
          exit 0
    fi

    # There should be no bogus failures below this line!
    log_info_msg "Setting up Linux console..."

    # Figure out if a framebuffer console is used
    [ -d /sys/class/graphics/fb0 ] && use_fb=1 || use_fb=0

    # Figure out the command to set the console into the
    # desired mode
    is_true "${UNICODE}" &&
        MODE_COMMAND="echo -en '\033%G' && kbd_mode -u" ||
        MODE_COMMAND="echo -en '\033%@\033(K' && kbd_mode -a"

    # On framebuffer consoles, font has to be set for each vt in
    # UTF-8 mode. This doesn't hurt in non-UTF-8 mode also.

    ! is_true "${use_fb}" || [ -z "${FONT}" ] ||
        MODE_COMMAND="${MODE_COMMAND} && setfont ${FONT}"

    # Apply that command to all consoles mentioned in
    # /etc/inittab. Important: in the UTF-8 mode this should
    # happen before setfont, otherwise a kernel bug will
    # show up and the unicode map of the font will not be
    # used.

    for TTY in `grep '^[^#].*respawn:/sbin/agetty' /etc/inittab |
        grep -o '\btty[[:digit:]]*\b'`
    do
        openvt -f -w -c ${TTY#tty} -- \
            /bin/sh -c "${MODE_COMMAND}" || failed=1
    done

    # Set the font (if not already set above) and the keymap
    [ "${use_fb}" == "1" ] || [ -z "${FONT}" ] || setfont $FONT || failed=1

    [ -z "${KEYMAP}" ] ||
        loadkeys ${KEYMAP} >/dev/null 2>&1 ||
        failed=1

    [ -z "${KEYMAP_CORRECTIONS}" ] ||
        loadkeys ${KEYMAP_CORRECTIONS} >/dev/null 2>&1 ||
        failed=1

    # Convert the keymap from $LEGACY_CHARSET to UTF-8
    [ -z "$LEGACY_CHARSET" ] ||
        dumpkeys -c "$LEGACY_CHARSET" | loadkeys -u >/dev/null 2>&1 ||
        failed=1

    # If any of the commands above failed, the trap at the
    # top would set $failed to 1
    ( exit $failed )
    evaluate_retval

    exit $failed
    ;;
```

```
    *)
        echo "Usage:  ${0} {start}"
        exit 1
        ;;
esac


# End console
```

D.14. /etc/rc.d/init.d/localnet

```
#!/bin/sh
########################################################################
# Begin localnet
#
# Description : Loopback device
#
# Authors     : Gerard Beekmans  - gerard AT linuxfromscratch D0T org
#               DJ Lucas - dj AT linuxfromscratch D0T org
# Update      : Bruce Dubbs - bdubbs AT linuxfromscratch D0T org
#
# Version     : LFS 7.0
#
########################################################################

### BEGIN INIT INFO
# Provides:            localnet
# Required-Start:      $local_fs
# Should-Start:
# Required-Stop:
# Should-Stop:
# Default-Start:       S
# Default-Stop:        0 6
# Short-Description:   Starts the local network.
# Description:         Sets the hostname of the machine and starts the
#                      loopback interface.
# X-LFS-Provided-By:   LFS
### END INIT INFO

. /lib/lsb/init-functions
[ -r /etc/sysconfig/network ] && . /etc/sysconfig/network

case "${1}" in
   start)
      log_info_msg "Bringing up the loopback interface..."
      ip addr add 127.0.0.1/8 label lo dev lo
      ip link set lo up
      evaluate_retval

      log_info_msg "Setting hostname to ${HOSTNAME}..."
      hostname ${HOSTNAME}
      evaluate_retval
      ;;

   stop)
      log_info_msg "Bringing down the loopback interface..."
```

```
      ip link set lo down
      evaluate_retval
      ;;

   restart)
      ${0} stop
      sleep 1
      ${0} start
      ;;

   status)
      echo "Hostname is: $(hostname)"
      ip link show lo
      ;;

   *)
      echo "Usage: ${0} {start|stop|restart|status}"
      exit 1
      ;;
esac

exit 0

# End localnet
```

D.15. /etc/rc.d/init.d/sysctl

```
#!/bin/sh
########################################################################
# Begin sysctl
#
# Description : File uses /etc/sysctl.conf to set kernel runtime
#               parameters
#
# Authors     : Nathan Coulson (nathan AT linuxfromscratch D0T org)
#               Matthew Burgress (matthew AT linuxfromscratch D0T org)
#               DJ Lucas - dj AT linuxfromscratch D0T org
# Update      : Bruce Dubbs - bdubbs AT linuxfromscratch D0T org
#
# Version     : LFS 7.0
#
########################################################################

### BEGIN INIT INFO
# Provides:            sysctl
# Required-Start:      mountvirtfs
# Should-Start:
# Required-Stop:
# Should-Stop:
# Default-Start:       S
# Default-Stop:
# Short-Description:   Makes changes to the proc filesystem
# Description:         Makes changes to the proc filesystem as defined in
#                      /etc/sysctl.conf.  See 'man sysctl(8)'.
# X-LFS-Provided-By:   LFS
### END INIT INFO
```

```
. /lib/lsb/init-functions

case "${1}" in
   start)
      if [ -f "/etc/sysctl.conf" ]; then
         log_info_msg "Setting kernel runtime parameters..."
         sysctl -q -p
         evaluate_retval
      fi
      ;;

   status)
      sysctl -a
      ;;

   *)
      echo "Usage: ${0} {start|status}"
      exit 1
      ;;
esac

exit 0

# End sysctl
```

D.16. /etc/rc.d/init.d/sysklogd

```
#!/bin/sh
########################################################################
# Begin sysklogd
#
# Description : Sysklogd loader
#
# Authors     : Gerard Beekmans - gerard AT linuxfromscratch D0T org
#               DJ Lucas - dj AT linuxfromscratch D0T org
# Update      : Bruce Dubbs - bdubbs AT linuxfromscratch D0T org
#
# Version     : LFS 7.0
#
########################################################################

### BEGIN INIT INFO
# Provides:            $syslog
# Required-Start:      localnet
# Should-Start:
# Required-Stop:       $local_fs sendsignals
# Should-Stop:
# Default-Start:       2 3 4 5
# Default-Stop:        0 1 6
# Short-Description:   Starts kernel and system log daemons.
# Description:         Starts kernel and system log daemons.
#                      /etc/fstab.
# X-LFS-Provided-By:   LFS
### END INIT INFO
```

```
. /lib/lsb/init-functions

case "${1}" in
   start)
      log_info_msg "Starting system log daemon..."
      parms=${SYSKLOGD_PARMS-'-m 0'}
      start_daemon /sbin/syslogd $parms
      evaluate_retval

      log_info_msg "Starting kernel log daemon..."
      start_daemon /sbin/klogd
      evaluate_retval
      ;;

   stop)
      log_info_msg "Stopping kernel log daemon..."
      killproc /sbin/klogd
      evaluate_retval

      log_info_msg "Stopping system log daemon..."
      killproc /sbin/syslogd
      evaluate_retval
      ;;

   reload)
      log_info_msg "Reloading system log daemon config file..."
      pid=`pidofproc syslogd`
      kill -HUP "${pid}"
      evaluate_retval
      ;;

   restart)
      ${0} stop
      sleep 1
      ${0} start
      ;;

   status)
      statusproc /sbin/syslogd
      statusproc klogd
      ;;

   *)
      echo "Usage: ${0} {start|stop|reload|restart|status}"
      exit 1
      ;;
esac

exit 0

# End sysklogd
```

D.17. /etc/rc.d/init.d/network

```
#!/bin/sh
########################################################################
```

```
# Begin network
#
# Description : Network Control Script
#
# Authors     : Gerard Beekmans - gerard AT linuxfromscratch DOT org
#               Nathan Coulson - nathan AT linuxfromscratch DOT org
#               Kevin P. Fleming - kpfleming@linuxfromscratch.org
#               DJ Lucas - dj AT linuxfromscratch DOT org
# Update      : Bruce Dubbs - bdubbs AT linuxfromscratch DOT org
#
# Version     : LFS 7.0
#
########################################################################

### BEGIN INIT INFO
# Provides:            $network
# Required-Start:      $local_fs swap localnet
# Should-Start:        $syslog
# Required-Stop:       $local_fs swap localnet
# Should-Stop:         $syslog
# Default-Start:       3 4 5
# Default-Stop:        0 1 2 6
# Short-Description:   Starts and configures network interfaces.
# Description:         Starts and configures network interfaces.
# X-LFS-Provided-By:   LFS
### END INIT INFO

case "${1}" in
   start)
      # Start all network interfaces
      for file in /etc/sysconfig/ifconfig.*
      do
         interface=${file##*/ifconfig.}

         # Skip if $file is * (because nothing was found)
         if [ "${interface}" = "*" ]
         then
            continue
         fi

         /sbin/ifup ${interface}
      done
      ;;

   stop)
      # Reverse list
      net_files=""
      for file in  /etc/sysconfig/ifconfig.*
      do
         net_files="${file} ${net_files}"
      done

      # Stop all network interfaces
      for file in ${net_files}
      do
         interface=${file##*/ifconfig.}
```

```
            # Skip if $file is * (because nothing was found)
            if [ "${interface}" = "*" ]
            then
                continue
            fi

            /sbin/ifdown ${interface}
        done
        ;;

    restart)
        ${0} stop
        sleep 1
        ${0} start
        ;;

    *)
        echo "Usage: ${0} {start|stop|restart}"
        exit 1
        ;;
esac

exit 0

# End network
```

D.18. /etc/rc.d/init.d/sendsignals

```
#!/bin/sh
########################################################################
# Begin sendsignals
#
# Description : Sendsignals Script
#
# Authors     : Gerard Beekmans - gerard AT linuxfromscratch D0T org
#               DJ Lucas - dj AT linuxfromscratch D0T org
# Update      : Bruce Dubbs - bdubbs AT linuxfromscratch D0T org
#
# Version     : LFS 7.0
#
########################################################################

### BEGIN INIT INFO
# Provides:            sendsignals
# Required-Start:
# Should-Start:
# Required-Stop:       $local_fs swap localnet
# Should-Stop:
# Default-Start:
# Default-Stop:        0 6
# Short-Description:   Attempts to kill remaining processes.
# Description:         Attempts to kill remaining processes.
# X-LFS-Provided-By:   LFS
### END INIT INFO

. /lib/lsb/init-functions
```

```
case "${1}" in
   stop)
      log_info_msg "Sending all processes the TERM signal..."
      killall5 -15
      error_value=${?}

      sleep ${KILLDELAY}

      if [ "${error_value}" = 0 -o "${error_value}" = 2 ]; then
         log_success_msg
      else
         log_failure_msg
      fi

      log_info_msg "Sending all processes the KILL signal..."
      killall5 -9
      error_value=${?}

      sleep ${KILLDELAY}

      if [ "${error_value}" = 0 -o "${error_value}" = 2 ]; then
         log_success_msg
      else
         log_failure_msg
      fi
      ;;

   *)
      echo "Usage: ${0} {stop}"
      exit 1
      ;;

esac

exit 0

# End sendsignals
```

D.19. /etc/rc.d/init.d/reboot

```
#!/bin/sh
########################################################################
# Begin reboot
#
# Description : Reboot Scripts
#
# Authors     : Gerard Beekmans - gerard AT linuxfromscratch D0T org
#               DJ Lucas - dj AT linuxfromscratch D0T org
# Update      : Bruce Dubbs - bdubbs AT linuxfromscratch D0T org
#
# Version     : LFS 7.0
#
########################################################################

### BEGIN INIT INFO
```

```
# Provides:            reboot
# Required-Start:
# Should-Start:
# Required-Stop:
# Should-Stop:
# Default-Start:       6
# Default-Stop:
# Short-Description:   Reboots the system.
# Description:         Reboots the System.
# X-LFS-Provided-By:   LFS
### END INIT INFO

. /lib/lsb/init-functions

case "${1}" in
   stop)
      log_info_msg "Restarting system..."
      reboot -d -f -i
      ;;

   *)
      echo "Usage: ${0} {stop}"
      exit 1
      ;;

esac

# End reboot
```

D.20. /etc/rc.d/init.d/halt

```
#!/bin/sh
########################################################################
# Begin halt
#
# Description : Halt Script
#
# Authors     : Gerard Beekmans - gerard AT linuxfromscratch D0T org
#               DJ Lucas - dj AT linuxfromscratch D0T org
# Update      : Bruce Dubbs - bdubbs AT linuxfromscratch D0T org
#
# Version     : LFS 7.0
#
########################################################################

### BEGIN INIT INFO
# Provides:            halt
# Required-Start:
# Should-Start:
# Required-Stop:
# Should-Stop:
# Default-Start:       0
# Default-Stop:
# Short-Description:   Halts the system.
# Description:         Halts the System.
# X-LFS-Provided-By:   LFS
```

```
### END INIT INFO

case "${1}" in
    stop)
        halt -d -f -i -p
        ;;

    *)
        echo "Usage: {stop}"
        exit 1
        ;;
esac

# End halt
```

D.21. /etc/rc.d/init.d/template

```
#!/bin/sh
########################################################################
# Begin scriptname
#
# Description :
#
# Authors     :
#
# Version     : LFS x.x
#
# Notes       :
#
########################################################################

### BEGIN INIT INFO
# Provides:            template
# Required-Start:
# Should-Start:
# Required-Stop:
# Should-Stop:
# Default-Start:
# Default-Stop:
# Short-Description:
# Description:
# X-LFS-Provided-By:
### END INIT INFO

. /lib/lsb/init-functions

case "${1}" in
    start)
        log_info_msg "Starting..."
        start_daemon fully_qualified_path
        ;;

    stop)
        log_info_msg "Stopping..."
        killproc fully_qualified_path
        ;;
```

```
   restart)
      ${0} stop
      sleep 1
      ${0} start
      ;;

   *)
      echo "Usage: ${0} {start|stop|restart}"
      exit 1
      ;;
esac


exit 0


# End scriptname
```

D.22. /etc/sysconfig/modules

```
##########################################################################
# Begin /etc/sysconfig/modules
#
# Description : Module auto-loading configuration
#
# Authors     :
#
# Version     : 00.00
#
# Notes       : The syntax of this file is as follows:
#           <module> [<arg1> <arg2> ...]
#
# Each module should be on it's own line, and any options that you want
# passed to the module should follow it.  The line deliminator is either
# a space or a tab.
##########################################################################


# End /etc/sysconfig/modules
```

D.23. /etc/sysconfig/createfiles

```
##########################################################################
# Begin /etc/sysconfig/createfiles
#
# Description : Createfiles script config file
#
# Authors     :
#
# Version     : 00.00
#
# Notes       : The syntax of this file is as follows:
#         if type is equal to "file" or "dir"
#           <filename> <type> <permissions> <user> <group>
#         if type is equal to "dev"
#            <filename> <type> <permissions> <user> <group> <devtype>
```

```
#                    <major> <minor>
#
#            <filename> is the name of the file which is to be created
#            <type> is either file, dir, or dev.
#                    file creates a new file
#                    dir creates a new directory
#                    dev creates a new device
#            <devtype> is either block, char or pipe
#                    block creates a block device
#                    char creates a character deivce
#                    pipe creates a pipe, this will ignore the <major> and
#             <minor> fields
#            <major> and <minor> are the major and minor numbers used for
#        the device.
##############################################################################


# End /etc/sysconfig/createfiles
```

D.24. /etc/sysconfig/udev-retry

```
##############################################################################
# Begin /etc/sysconfig/udev_retry
#
# Description : udev_retry script configuration
#
# Authors     :
#
# Version     : 00.00
#
# Notes       : Each subsystem that may need to be re-triggered after mountfs
#               runs should be listed in this file.  Probable subsystems to be
#               listed here are rtc (due to /var/lib/hwclock/adjtime) and sound
#               (due to both /var/lib/alsa/asound.state and /usr/sbin/alsactl).
#               Entries are whitespace-separated.
##############################################################################


rtc


# End /etc/sysconfig/udev_retry
```

D.25. /sbin/ifup

```
#!/bin/sh
##############################################################################
# Begin /sbin/ifup
#
# Description : Interface Up
#
# Authors     : Nathan Coulson - nathan AT linuxfromscratch D0T org
#               Kevin P. Fleming - kpfleming@linuxfromscratch.org
# Update      : Bruce Dubbs - bdubbs AT linuxfromscratch D0T org
#
# Version     : LFS 7.2
#
```

```
# Notes        : The IFCONFIG variable is passed to the SERVICE script
#                in the /lib/services directory, to indicate what file the
#                service should source to get interface specifications.
#
##########################################################################

up()
{
  if ip link show $1 > /dev/null 2>&1; then
     link_status=`ip link show $1`

     if [ -n "${link_status}" ]; then
        if ! echo "${link_status}" | grep -q UP; then
           ip link set $1 up
        fi
     fi

  else
     log_failure_msg "\nInterface ${IFACE} doesn't exist."
     exit 1
  fi
}

RELEASE="7.2"

USAGE="Usage: $0 [ -hV ] [--help] [--version] interface"
VERSTR="LFS ifup, version ${RELEASE}"

while [ $# -gt 0 ]; do
   case "$1" in
      --help | -h)    help="y"; break ;;

      --version | -V)  echo "${VERSTR}"; exit 0 ;;

      -*)             echo "ifup: ${1}: invalid option" >&2
                      echo "${USAGE}" >& 2
                      exit 2 ;;

      *)              break ;;
   esac
done

if [ -n "$help" ]; then
   echo "${VERSTR}"
   echo "${USAGE}"
   echo
   cat << HERE_EOF
ifup is used to bring up a network interface.  The interface
parameter, e.g. eth0 or eth0:2, must match the trailing part of the
interface specifications file, e.g. /etc/sysconfig/ifconfig.eth0:2.

HERE_EOF
   exit 0
fi

file=/etc/sysconfig/ifconfig.${1}
```

```
# Skip backup files
[ "${file}" = "${file%""~""}" ] || exit 0


. /lib/lsb/init-functions


log_info_msg "Bringing up the ${1} interface... "


if [ ! -r "${file}" ]; then
   log_failure_msg2 "${file} is missing or cannot be accessed."
   exit 1
fi


. $file


if [ "$IFACE" = "" ]; then
   log_failure_msg2 "${file} does not define an interface [IFACE]."
   exit 1
fi


# Do not process this service if started by boot, and ONBOOT
# is not set to yes
if [ "${IN_BOOT}" = "1" -a "${ONBOOT}" != "yes" ]; then
   log_info_msg2 "skipped"
   exit 0
fi


for S in ${SERVICE}; do
  if [ ! -x "/lib/services/${S}" ]; then
    MSG="\nUnable to process ${file}.  Either "
    MSG="${MSG}the SERVICE '${S} was not present "
    MSG="${MSG}or cannot be executed."
    log_failure_msg "$MSG"
    exit 1
  fi
done


# Create/configure the interface
for S in ${SERVICE}; do
  IFCONFIG=${file} /lib/services/${S} ${IFACE} up
done


# Bring up the interface and any components
for I in $IFACE $INTERFACE_COMPONENTS; do up $I; done


# Set MTU if requested. Check if MTU has a "good" value.
if test -n "${MTU}"; then
   if [[ ${MTU} =~ ^[0-9]+$ ]] && [[ $MTU -ge 68 ]] ; then
       for I in $IFACE $INTERFACE_COMPONENTS; do
           ip link set dev $I mtu $MTU;
       done
   else
       log_info_msg2 "Invalid MTU $MTU"
   fi
fi


# Set the route default gateway if requested
if [ -n "${GATEWAY}" ]; then
```

```
    if ip route | grep -q default; then
        log_warning_msg "\nGateway already setup; skipping."
    else
        log_info_msg "Setting up default gateway..."
        ip route add default via ${GATEWAY} dev ${IFACE}
        evaluate_retval
    fi
fi


# End /sbin/ifup
```

D.26. /sbin/ifdown

```
#!/bin/bash
########################################################################
# Begin /sbin/ifdown
#
# Description : Interface Down
#
# Authors     : Nathan Coulson - nathan AT linuxfromscratch D0T org
#               Kevin P. Fleming - kpfleming@linuxfromscratch.org
# Update      : Bruce Dubbs - bdubbs AT linuxfromscratch D0T org
#
# Version     : LFS 7.0
#
# Notes       : the IFCONFIG variable is passed to the scripts found
#               in the /lib/services directory, to indicate what file the
#               service should source to get interface specifications.
#
########################################################################

RELEASE="7.0"

USAGE="Usage: $0 [ -hV ] [--help] [--version] interface"
VERSTR="LFS ifdown, version ${RELEASE}"

while [ $# -gt 0 ]; do
    case "$1" in
        --help | -h)    help="y"; break ;;

        --version | -V)  echo "${VERSTR}"; exit 0 ;;

        -*)              echo "ifup: ${1}: invalid option" >&2
                         echo "${USAGE}" >& 2
                         exit 2 ;;

        *)               break ;;
    esac
done


if [ -n "$help" ]; then
    echo "${VERSTR}"
    echo "${USAGE}"
    echo
    cat << HERE_EOF
ifdown is used to bring down a network interface.  The interface
```

```
parameter, e.g. eth0 or eth0:2, must match the trailing part of the
interface specifications file, e.g. /etc/sysconfig/ifconfig.eth0:2.


HERE_EOF
   exit 0
fi


file=/etc/sysconfig/ifconfig.${1}


# Skip backup files
[ "${file}" = "${file%""~""}" ] || exit 0


. /lib/lsb/init-functions


if [ ! -r "${file}" ]; then
   log_warning_msg "${file} is missing or cannot be accessed."
   exit 1
fi


. ${file}


if [ "$IFACE" = "" ]; then
   log_failure_msg "${file} does not define an interface [IFACE]."
   exit 1
fi


# We only need to first service to bring down the interface
S=`echo ${SERVICE} | cut -f1 -d" "`


if ip link show ${IFACE} > /dev/null 2>&1; then
   if [ -n "${S}" -a -x "/lib/services/${S}" ]; then
     IFCONFIG=${file} /lib/services/${S} ${IFACE} down
   else
     MSG="Unable to process ${file}.  Either "
     MSG="${MSG}the SERVICE variable was not set "
     MSG="${MSG}or the specified service cannot be executed."
     log_failure_msg "$MSG"
     exit 1
  fi
else
   log_warning_msg "Interface ${1} doesn't exist."
fi


# Leave the interface up if there are additional interfaces in the device
link_status=`ip link show ${IFACE} 2>/dev/null`


if [ -n "${link_status}" ]; then
   if [ "$(echo "${link_status}" | grep UP)" != "" ]; then
       if [ "$(ip addr show ${IFACE} | grep 'inet ')" == ""  ]; then
           log_info_msg "Bringing down the ${IFACE} interface..."
           ip link set ${IFACE} down
           evaluate_retval
       fi
   fi
fi


# End /sbin/ifdown
```

D.27. /lib/services/ipv4-static

```
#!/bin/sh
########################################################################
# Begin /lib/services/ipv4-static
#
# Description : IPV4 Static Boot Script
#
# Authors     : Nathan Coulson - nathan AT linuxfromscratch D0T org
#               Kevin P. Fleming - kpfleming@linuxfromscratch.org
# Update      : Bruce Dubbs - bdubbs AT linuxfromscratch D0T org
#
# Version     : LFS 7.0
#
########################################################################

. /lib/lsb/init-functions
. ${IFCONFIG}

if [ -z "${IP}" ]; then
    log_failure_msg "\nIP variable missing from ${IFCONFIG}, cannot continue."
    exit 1
fi

if [ -z "${PREFIX}" -a -z "${PEER}" ]; then
    log_warning_msg "\nPREFIX variable missing from ${IFCONFIG}, assuming 24."
    PREFIX=24
    args="${args} ${IP}/${PREFIX}"

elif [ -n "${PREFIX}" -a -n "${PEER}" ]; then
    log_failure_msg "\nPREFIX and PEER both specified in ${IFCONFIG}, cannot continue.
    exit 1

elif [ -n "${PREFIX}" ]; then
    args="${args} ${IP}/${PREFIX}"

elif [ -n "${PEER}" ]; then
    args="${args} ${IP} peer ${PEER}"
fi

if [ -n "${BROADCAST}" ]; then
    args="${args} broadcast ${BROADCAST}"
fi

case "${2}" in
    up)
        if [ "$(ip addr show ${1} 2>/dev/null | grep ${IP}/)" = "" ]; then

            # Cosmetic output not needed for multiple services
            if ! $(echo ${SERVICE} | grep -q " "); then
              log_info_msg2 "\n" # Terminate the previous message
            fi

            log_info_msg "Adding IPv4 address ${IP} to the ${1} interface..."
            ip addr add ${args} dev ${1}
            evaluate_retval
```

```
        else
            log_warning_msg "Cannot add IPv4 address ${IP} to ${1}.  Already present."
        fi
    ;;

    down)
        if [ "$(ip addr show ${1} 2>/dev/null | grep ${IP}/)" != "" ]; then
            log_info_msg "Removing IPv4 address ${IP} from the ${1} interface..."
            ip addr del ${args} dev ${1}
            evaluate_retval
        fi

        if [ -n "${GATEWAY}" ]; then
            # Only remove the gateway if there are no remaining ipv4 addresses
            if [ "$(ip addr show ${1} 2>/dev/null | grep 'inet ')" != "" ]; then
                log_info_msg "Removing default gateway..."
                ip route del default
                evaluate_retval
            fi
        fi
    ;;

    *)
        echo "Usage: ${0} [interface] {up|down}"
        exit 1
    ;;
esac

# End /lib/services/ipv4-static
```

D.28. /lib/services/ipv4-static-route

```
#!/bin/sh
########################################################################
# Begin /lib/services/ipv4-static-route
#
# Description : IPV4 Static Route Script
#
# Authors     : Kevin P. Fleming - kpfleming@linuxfromscratch.org
#               DJ Lucas - dj AT linuxfromscratch D0T org
# Update      : Bruce Dubbs - bdubbs AT linuxfromscratch D0T org
#
# Version     : LFS 7.0
#
########################################################################

. /lib/lsb/init-functions
. ${IFCONFIG}

case "${TYPE}" in
    ("" | "network")
        need_ip=1
        need_gateway=1
    ;;

    ("default")
```

```
            need_gateway=1
            args="${args} default"
            desc="default"
        ;;

        ("host")
            need_ip=1
        ;;

        ("unreachable")
            need_ip=1
            args="${args} unreachable"
            desc="unreachable "
        ;;

        (*)
            log_failure_msg "Unknown route type (${TYPE}) in ${IFCONFIG}, cannot continue.'
            exit 1
        ;;
esac

if [ -n "${need_ip}" ]; then
    if [ -z "${IP}" ]; then
        log_failure_msg "IP variable missing from ${IFCONFIG}, cannot continue."
        exit 1
    fi

    if [ -z "${PREFIX}" ]; then
        log_failure_msg "PREFIX variable missing from ${IFCONFIG}, cannot continue."
        exit 1
    fi

    args="${args} ${IP}/${PREFIX}"
    desc="${desc}${IP}/${PREFIX}"
fi

if [ -n "${need_gateway}" ]; then
    if [ -z "${GATEWAY}" ]; then
        log_failure_msg "GATEWAY variable missing from ${IFCONFIG}, cannot continue."
        exit 1
    fi
    args="${args} via ${GATEWAY}"
fi

if [ -n "${SOURCE}" ]; then
        args="${args} src ${SOURCE}"
fi

case "${2}" in
    up)
        log_info_msg "Adding '${desc}' route to the ${1} interface..."
        ip route add ${args} dev ${1}
        evaluate_retval
    ;;

    down)
        log_info_msg "Removing '${desc}' route from the ${1} interface..."
```

```
        ip route del ${args} dev ${1}
        evaluate_retval
    ;;


    *)
        echo "Usage: ${0} [interface] {up|down}"
        exit 1
    ;;
esac


# End /lib/services/ipv4-static-route
```

Appendix E. Udev configuration rules

The rules from udev-lfs-197-2.tar.bz2 in this appendix are listed for convenience. Installation is normally done via instructions in [Section 6.61, "Udev-197 (Extracted from systemd-197)"](#).

E.1. 55-lfs.rules

```
# /etc/udev/rules.d/55-lfs.rules: Rule definitions for LFS.

# Core kernel devices

# This causes the system clock to be set as soon as /dev/rtc becomes available.
SUBSYSTEM=="rtc", ACTION=="add", MODE="0644", RUN+="/etc/rc.d/init.d/setclock start"
KERNEL=="rtc", ACTION=="add", MODE="0644", RUN+="/etc/rc.d/init.d/setclock start"

# Comms devices

KERNEL=="ippp[0-9]*",       GROUP="dialout"
KERNEL=="isdn[0-9]*",       GROUP="dialout"
KERNEL=="isdnctrl[0-9]*",   GROUP="dialout"
KERNEL=="dcbri[0-9]*",      GROUP="dialout"
```

Appendix F. LFS Licenses

This book is licensed under the Creative Commons Attribution-NonCommercial-ShareAlike 2.0 License.

Computer instructions may be extracted from the book under the MIT License.

F.1. Creative Commons License

Creative Commons Legal Code

Attribution-NonCommercial-ShareAlike 2.0

Important

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.

F.2. The MIT License

Index

Packages

Binutils:	Binutils-2.23.1
tools, pass 1:	Binutils-2.23.1 - Pass 1
tools, pass 2:	Binutils-2.23.1 - Pass 2
Bison:	Bison-2.7
Bootscripts:	LFS-Bootscripts-20130123
usage:	How Do These Bootscripts Work?
Bzip2:	Bzip2-1.0.6
tools:	Bzip2-1.0.6
Check:	Check-0.9.9
Coreutils:	Coreutils-8.21
tools:	Coreutils-8.21
DejaGNU:	DejaGNU-1.5
Diffutils:	Diffutils-3.2
tools:	Diffutils-3.2
E2fsprogs:	E2fsprogs-1.42.7
Expect:	Expect-5.45
File:	File-5.13
tools:	File-5.13
Findutils:	Findutils-4.4.2
tools:	Findutils-4.4.2
Flex:	Flex-2.5.37
Gawk:	Gawk-4.0.2
tools:	Gawk-4.0.2
GCC:	GCC-4.7.2
tools, pass 1:	GCC-4.7.2 - Pass 1
tools, pass 2:	GCC-4.7.2 - Pass 2
GDBM:	GDBM-1.10
Gettext:	Gettext-0.18.2
tools:	Gettext-0.18.2
Glibc:	Glibc-2.17
tools:	Glibc-2.17
GMP:	GMP-5.1.1
Grep:	Grep-2.14
tools:	Grep-2.14
Groff:	Groff-1.22.2
GRUB:	GRUB-2.00
Gzip:	Gzip-1.5
tools:	Gzip-1.5
Iana-Etc:	Iana-Etc-2.30
Inetutils:	Inetutils-1.9.1
IPRoute2:	IPRoute2-3.8.0
Kbd:	Kbd-1.15.5
Kmod:	Kmod-12
Less:	Less-451
Libpipeline:	Libpipeline-1.2.2
Libtool:	Libtool-2.4.2
Linux:	Linux-3.8.1
API headers:	Linux-3.8.1 API Headers
tools, API headers:	Linux-3.8.1 API Headers
M4:	M4-1.4.16
tools:	M4-1.4.16
Make:	Make-3.82
tools:	Make-3.82
Man-DB:	Man-DB-2.6.3
Man-pages:	Man-pages-3.47
MPC:	MPC-1.0.1
MPFR:	MPFR-3.1.1

Ncurses:	Ncurses-5.9
tools:	Ncurses-5.9
Patch:	Patch-2.7.1
tools:	Patch-2.7.1
Perl:	Perl-5.16.2
tools:	Perl-5.16.2
pkgconfig:	Pkg-config-0.28
Procps-ng:	Procps-ng-3.3.6
Psmisc:	Psmisc-22.20
rc.site:	The rc.site File
Readline:	Readline-6.2
Sed:	Sed-4.2.2
tools:	Sed-4.2.2
Shadow:	Shadow-4.1.5.1
configuring:	Configuring Shadow
Sysklogd:	Sysklogd-1.5
configuring:	Configuring Sysklogd
Sysvinit:	Sysvinit-2.88dsf
configuring:	Configuring Sysvinit
Tar:	Tar-1.26
tools:	Tar-1.26
Tcl:	Tcl-8.6.0
Texinfo:	Texinfo-5.0
tools:	Texinfo-5.0
Udev:	Udev-197 (Extracted from systemd-197)
usage:	Device and Module Handling on an LFS System
Util-linux:	Util-linux-2.22.2
Vim:	Vim-7.3
xz:	Xz-5.0.4
tools:	Xz-5.0.4
Zlib:	Zlib-1.2.7

Programs

a2p:	Perl-5.16.2 -- description
accessdb:	Man-DB-2.6.3 -- description
acinstall:	Automake-1.13.1 -- description
aclocal:	Automake-1.13.1 -- description
aclocal-1.13:	Automake-1.13.1 -- description
addftinfo:	Groff-1.22.2 -- description
addpart:	Util-linux-2.22.2 -- description
addr2line:	Binutils-2.23.1 -- description
afmtodit:	Groff-1.22.2 -- description
agetty:	Util-linux-2.22.2 -- description
apropos:	Man-DB-2.6.3 -- description
ar:	Binutils-2.23.1 -- description
as:	Binutils-2.23.1 -- description
ata_id:	Udev-197 (Extracted from systemd-197) -- description
autoconf:	Autoconf-2.69 -- description
autoheader:	Autoconf-2.69 -- description
autom4te:	Autoconf-2.69 -- description
automake:	Automake-1.13.1 -- description
automake-1.13:	Automake-1.13.1 -- description
autopoint:	Gettext-0.18.2 -- description
autoreconf:	Autoconf-2.69 -- description
autoscan:	Autoconf-2.69 -- description
autoupdate:	Autoconf-2.69 -- description

awk:	Gawk-4.0.2 -- description
badblocks:	E2fsprogs-1.42.7 -- description
base64:	Coreutils-8.21 -- description
basename:	Coreutils-8.21 -- description
bash:	Bash-4.2 -- description
bashbug:	Bash-4.2 -- description
bigram:	Findutils-4.4.2 -- description
bison:	Bison-2.7 -- description
blkid:	Util-linux-2.22.2 -- description
blockdev:	Util-linux-2.22.2 -- description
bootlogd:	Sysvinit-2.88dsf -- description
bridge:	IPRoute2-3.8.0 -- description
bunzip2:	Bzip2-1.0.6 -- description
bzcat:	Bzip2-1.0.6 -- description
bzcmp:	Bzip2-1.0.6 -- description
bzdiff:	Bzip2-1.0.6 -- description
bzegrep:	Bzip2-1.0.6 -- description
bzfgrep:	Bzip2-1.0.6 -- description
bzgrep:	Bzip2-1.0.6 -- description
bzip2:	Bzip2-1.0.6 -- description
bzip2recover:	Bzip2-1.0.6 -- description
bzless:	Bzip2-1.0.6 -- description
bzmore:	Bzip2-1.0.6 -- description
c++:	GCC-4.7.2 -- description
c++filt:	Binutils-2.23.1 -- description
c2ph:	Perl-5.16.2 -- description
cal:	Util-linux-2.22.2 -- description
captoinfo:	Ncurses-5.9 -- description
cat:	Coreutils-8.21 -- description
catchsegv:	Glibc-2.17 -- description
catman:	Man-DB-2.6.3 -- description
cc:	GCC-4.7.2 -- description
cdrom_id:	Udev-197 (Extracted from systemd-197) -- description
cfdisk:	Util-linux-2.22.2 -- description
chage:	Shadow-4.1.5.1 -- description
chattr:	E2fsprogs-1.42.7 -- description
chcon:	Coreutils-8.21 -- description
chcpu:	Util-linux-2.22.2 -- description
checkmk:	Check-0.9.9 -- description
chem:	Groff-1.22.2 -- description
chfn:	Shadow-4.1.5.1 -- description
chgpasswd:	Shadow-4.1.5.1 -- description
chgrp:	Coreutils-8.21 -- description
chmod:	Coreutils-8.21 -- description
chown:	Coreutils-8.21 -- description
chpasswd:	Shadow-4.1.5.1 -- description
chroot:	Coreutils-8.21 -- description
chrt:	Util-linux-2.22.2 -- description
chsh:	Shadow-4.1.5.1 -- description
chvt:	Kbd-1.15.5 -- description
cksum:	Coreutils-8.21 -- description
clear:	Ncurses-5.9 -- description
cmp:	Diffutils-3.2 -- description
code:	Findutils-4.4.2 -- description
col:	Util-linux-2.22.2 -- description
colcrt:	Util-linux-2.22.2 -- description
collect:	Udev-197 (Extracted from systemd-197) -- description

colrm:	Util-linux-2.22.2 -- description
column:	Util-linux-2.22.2 -- description
comm:	Coreutils-8.21 -- description
compile:	Automake-1.13.1 -- description
compile_et:	E2fsprogs-1.42.7 -- description
config.charset:	Gettext-0.18.2 -- description
config.guess:	Automake-1.13.1 -- description
config.rpath:	Gettext-0.18.2 -- description
config.sub:	Automake-1.13.1 -- description
config_data:	Perl-5.16.2 -- description
corelist:	Perl-5.16.2 -- description
cp:	Coreutils-8.21 -- description
cpan:	Perl-5.16.2 -- description
cpan2dist:	Perl-5.16.2 -- description
cpanp:	Perl-5.16.2 -- description
cpanp-run-perl:	Perl-5.16.2 -- description
cpp:	GCC-4.7.2 -- description
csplit:	Coreutils-8.21 -- description
ctrlaltdel:	Util-linux-2.22.2 -- description
ctstat:	IPRoute2-3.8.0 -- description
cut:	Coreutils-8.21 -- description
cytune:	Util-linux-2.22.2 -- description
date:	Coreutils-8.21 -- description
dd:	Coreutils-8.21 -- description
deallocvt:	Kbd-1.15.5 -- description
debugfs:	E2fsprogs-1.42.7 -- description
delpart:	Util-linux-2.22.2 -- description
depcomp:	Automake-1.13.1 -- description
depmod:	Kmod-12 -- description
df:	Coreutils-8.21 -- description
dgawk:	Gawk-4.0.2 -- description
diff:	Diffutils-3.2 -- description
diff3:	Diffutils-3.2 -- description
dir:	Coreutils-8.21 -- description
dircolors:	Coreutils-8.21 -- description
dirname:	Coreutils-8.21 -- description
dmesg:	Util-linux-2.22.2 -- description
du:	Coreutils-8.21 -- description
dumpe2fs:	E2fsprogs-1.42.7 -- description
dumpkeys:	Kbd-1.15.5 -- description
e2freefrag:	E2fsprogs-1.42.7 -- description
e2fsck:	E2fsprogs-1.42.7 -- description
e2image:	E2fsprogs-1.42.7 -- description
e2initrd_helper:	E2fsprogs-1.42.7 -- description
e2label:	E2fsprogs-1.42.7 -- description
e2undo:	E2fsprogs-1.42.7 -- description
e4defrag:	E2fsprogs-1.42.7 -- description
echo:	Coreutils-8.21 -- description
egrep:	Grep-2.14 -- description
eject:	Util-linux-2.22.2 -- description
elfedit:	Binutils-2.23.1 -- description
elisp-comp:	Automake-1.13.1 -- description
enc2xs:	Perl-5.16.2 -- description
env:	Coreutils-8.21 -- description
envsubst:	Gettext-0.18.2 -- description
eqn:	Groff-1.22.2 -- description
eqn2graph:	Groff-1.22.2 -- description

ex:	Vim-7.3 -- description
expand:	Coreutils-8.21 -- description
expect:	Expect-5.45 -- description
expiry:	Shadow-4.1.5.1 -- description
expr:	Coreutils-8.21 -- description
factor:	Coreutils-8.21 -- description
faillog:	Shadow-4.1.5.1 -- description
fallocate:	Util-linux-2.22.2 -- description
false:	Coreutils-8.21 -- description
fdformat:	Util-linux-2.22.2 -- description
fdisk:	Util-linux-2.22.2 -- description
fgconsole:	Kbd-1.15.5 -- description
fgrep:	Grep-2.14 -- description
file:	File-5.13 -- description
filefrag:	E2fsprogs-1.42.7 -- description
find:	Findutils-4.4.2 -- description
find2perl:	Perl-5.16.2 -- description
findfs:	Util-linux-2.22.2 -- description
findmnt:	Util-linux-2.22.2 -- description
flex:	Flex-2.5.37 -- description
flex++:	Flex-2.5.37 -- description
flock:	Util-linux-2.22.2 -- description
fmt:	Coreutils-8.21 -- description
fold:	Coreutils-8.21 -- description
frcode:	Findutils-4.4.2 -- description
free:	Procps-ng-3.3.6 -- description
fsck:	Util-linux-2.22.2 -- description
fsck.cramfs:	Util-linux-2.22.2 -- description
fsck.ext2:	E2fsprogs-1.42.7 -- description
fsck.ext3:	E2fsprogs-1.42.7 -- description
fsck.ext4:	E2fsprogs-1.42.7 -- description
fsck.ext4dev:	E2fsprogs-1.42.7 -- description
fsck.minix:	Util-linux-2.22.2 -- description
fsfreeze:	Util-linux-2.22.2 -- description
fstab-decode:	Sysvinit-2.88dsf -- description
fstrim:	Util-linux-2.22.2 -- description
ftp:	Inetutils-1.9.1 -- description
fuser:	Psmisc-22.20 -- description
g++:	GCC-4.7.2 -- description
gawk:	Gawk-4.0.2 -- description
gawk-4.0.2:	Gawk-4.0.2 -- description
gcc:	GCC-4.7.2 -- description
gc-ar:	GCC-4.7.2 -- description
gc-nm:	GCC-4.7.2 -- description
gc-ranlib:	GCC-4.7.2 -- description
gccbug:	GCC-4.7.2 -- description
gcov:	GCC-4.7.2 -- description
gdiffmk:	Groff-1.22.2 -- description
gencat:	Glibc-2.17 -- description
genl:	IPRoute2-3.8.0 -- description
geqn:	Groff-1.22.2 -- description
getconf:	Glibc-2.17 -- description
getent:	Glibc-2.17 -- description
getkeycodes:	Kbd-1.15.5 -- description
getopt:	Util-linux-2.22.2 -- description
gettext:	Gettext-0.18.2 -- description
gettext.sh:	Gettext-0.18.2 -- description

gettextize:	Gettext-0.18.2 -- description
gpasswd:	Shadow-4.1.5.1 -- description
gprof:	Binutils-2.23.1 -- description
grap2graph:	Groff-1.22.2 -- description
grcat:	Gawk-4.0.2 -- description
grep:	Grep-2.14 -- description
grn:	Groff-1.22.2 -- description
grodvi:	Groff-1.22.2 -- description
groff:	Groff-1.22.2 -- description
groffer:	Groff-1.22.2 -- description
grog:	Groff-1.22.2 -- description
grolbp:	Groff-1.22.2 -- description
grolj4:	Groff-1.22.2 -- description
grops:	Groff-1.22.2 -- description
grotty:	Groff-1.22.2 -- description
groupadd:	Shadow-4.1.5.1 -- description
groupdel:	Shadow-4.1.5.1 -- description
groupmems:	Shadow-4.1.5.1 -- description
groupmod:	Shadow-4.1.5.1 -- description
groups:	Coreutils-8.21 -- description
grpck:	Shadow-4.1.5.1 -- description
grpconv:	Shadow-4.1.5.1 -- description
grpunconv:	Shadow-4.1.5.1 -- description
grub-bios-setup:	GRUB-2.00 -- description
grub-editenv:	GRUB-2.00 -- description
grub-fstest:	GRUB-2.00 -- description
grub-install:	GRUB-2.00 -- description
grub-kbdcomp:	GRUB-2.00 -- description
grub-menulst2cfg:	GRUB-2.00 -- description
grub-mkconfig:	GRUB-2.00 -- description
grub-mkimage:	GRUB-2.00 -- description
grub-mklayout:	GRUB-2.00 -- description
grub-mknetdir:	GRUB-2.00 -- description
grub-mkpasswd-pbkdf2:	GRUB-2.00 -- description
grub-mkrelpath:	GRUB-2.00 -- description
grub-mkrescue:	GRUB-2.00 -- description
grub-mkstandalone:	GRUB-2.00 -- description
grub-ofpathname:	GRUB-2.00 -- description
grub-probe:	GRUB-2.00 -- description
grub-reboot:	GRUB-2.00 -- description
grub-script-check:	GRUB-2.00 -- description
grub-set-default:	GRUB-2.00 -- description
grub-setup:	GRUB-2.00 -- description
gtbl:	Groff-1.22.2 -- description
gunzip:	Gzip-1.5 -- description
gzexe:	Gzip-1.5 -- description
gzip:	Gzip-1.5 -- description
h2ph:	Perl-5.16.2 -- description
h2xs:	Perl-5.16.2 -- description
halt:	Sysvinit-2.88dsf -- description
head:	Coreutils-8.21 -- description
hexdump:	Util-linux-2.22.2 -- description
hostid:	Coreutils-8.21 -- description
hostname:	Inetutils-1.9.1 -- description
hostname:	Gettext-0.18.2 -- description
hpftodit:	Groff-1.22.2 -- description
hwclock:	Util-linux-2.22.2 -- description

i386:	Util-linux-2.22.2 -- description
iconv:	Glibc-2.17 -- description
iconvconfig:	Glibc-2.17 -- description
id:	Coreutils-8.21 -- description
ifcfg:	IPRoute2-3.8.0 -- description
ifnames:	Autoconf-2.69 -- description
ifstat:	IPRoute2-3.8.0 -- description
igawk:	Gawk-4.0.2 -- description
indxbib:	Groff-1.22.2 -- description
info:	Texinfo-5.0 -- description
infocmp:	Ncurses-5.9 -- description
infokey:	Texinfo-5.0 -- description
infotocap:	Ncurses-5.9 -- description
init:	Sysvinit-2.88dsf -- description
insmod:	Kmod-12 -- description
install:	Coreutils-8.21 -- description
install-info:	Texinfo-5.0 -- description
install-sh:	Automake-1.13.1 -- description
instmodsh:	Perl-5.16.2 -- description
ionice:	Util-linux-2.22.2 -- description
ip:	IPRoute2-3.8.0 -- description
ipcmk:	Util-linux-2.22.2 -- description
ipcrm:	Util-linux-2.22.2 -- description
ipcs:	Util-linux-2.22.2 -- description
isosize:	Util-linux-2.22.2 -- description
join:	Coreutils-8.21 -- description
json_pp:	Perl-5.16.2 -- description
kbdinfo:	Kbd-1.15.5 -- description
kbdrate:	Kbd-1.15.5 -- description
kbd_mode:	Kbd-1.15.5 -- description
kill:	Util-linux-2.22.2 -- description
killall:	Psmisc-22.20 -- description
killall5:	Sysvinit-2.88dsf -- description
klogd:	Sysklogd-1.5 -- description
kmod:	Kmod-12 -- description
last:	Sysvinit-2.88dsf -- description
lastb:	Sysvinit-2.88dsf -- description
lastlog:	Shadow-4.1.5.1 -- description
ld:	Binutils-2.23.1 -- description
ld.bfd:	Binutils-2.23.1 -- description
ldattach:	Util-linux-2.22.2 -- description
ldconfig:	Glibc-2.17 -- description
ldd:	Glibc-2.17 -- description
lddlibc4:	Glibc-2.17 -- description
less:	Less-451 -- description
lessecho:	Less-451 -- description
lesskey:	Less-451 -- description
lex:	Flex-2.5.37 -- description
lexgrog:	Man-DB-2.6.3 -- description
lfskernel-3.8.1:	Linux-3.8.1 -- description
libnetcfg:	Perl-5.16.2 -- description
libtool:	Libtool-2.4.2 -- description
libtoolize:	Libtool-2.4.2 -- description
link:	Coreutils-8.21 -- description
linux32:	Util-linux-2.22.2 -- description
linux64:	Util-linux-2.22.2 -- description
lkbib:	Groff-1.22.2 -- description

ln:	Coreutils-8.21 -- description
lnstat:	IPRoute2-3.8.0 -- description
loadkeys:	Kbd-1.15.5 -- description
loadunimap:	Kbd-1.15.5 -- description
locale:	Glibc-2.17 -- description
localedef:	Glibc-2.17 -- description
locate:	Findutils-4.4.2 -- description
logger:	Util-linux-2.22.2 -- description
login:	Shadow-4.1.5.1 -- description
logname:	Coreutils-8.21 -- description
logoutd:	Shadow-4.1.5.1 -- description
logsave:	E2fsprogs-1.42.7 -- description
look:	Util-linux-2.22.2 -- description
lookbib:	Groff-1.22.2 -- description
losetup:	Util-linux-2.22.2 -- description
ls:	Coreutils-8.21 -- description
lsattr:	E2fsprogs-1.42.7 -- description
lsblk:	Util-linux-2.22.2 -- description
lscpu:	Util-linux-2.22.2 -- description
lslocks:	Util-linux-2.22.2 -- description
lsmod:	Kmod-12 -- description
lzcat:	Xz-5.0.4 -- description
lzcmp:	Xz-5.0.4 -- description
lzdiff:	Xz-5.0.4 -- description
lzegrep:	Xz-5.0.4 -- description
lzfgrep:	Xz-5.0.4 -- description
lzgrep:	Xz-5.0.4 -- description
lzless:	Xz-5.0.4 -- description
lzma:	Xz-5.0.4 -- description
lzmadec:	Xz-5.0.4 -- description
lzmainfo:	Xz-5.0.4 -- description
lzmore:	Xz-5.0.4 -- description
m4:	M4-1.4.16 -- description
make:	Make-3.82 -- description
makedb:	Glibc-2.17 -- description
makeinfo:	Texinfo-5.0 -- description
man:	Man-DB-2.6.3 -- description
mandb:	Man-DB-2.6.3 -- description
manpath:	Man-DB-2.6.3 -- description
mapscrn:	Kbd-1.15.5 -- description
mcookie:	Util-linux-2.22.2 -- description
md5sum:	Coreutils-8.21 -- description
mdate-sh:	Automake-1.13.1 -- description
mesg:	Sysvinit-2.88dsf -- description
missing:	Automake-1.13.1 -- description
mkdir:	Coreutils-8.21 -- description
mke2fs:	E2fsprogs-1.42.7 -- description
mkfifo:	Coreutils-8.21 -- description
mkfs:	Util-linux-2.22.2 -- description
mkfs.bfs:	Util-linux-2.22.2 -- description
mkfs.cramfs:	Util-linux-2.22.2 -- description
mkfs.ext2:	E2fsprogs-1.42.7 -- description
mkfs.ext3:	E2fsprogs-1.42.7 -- description
mkfs.ext4:	E2fsprogs-1.42.7 -- description
mkfs.ext4dev:	E2fsprogs-1.42.7 -- description
mkfs.minix:	Util-linux-2.22.2 -- description
mkinstalldirs:	Automake-1.13.1 -- description

mklost+found:	E2fsprogs-1.42.7 -- description
mknod:	Coreutils-8.21 -- description
mkswap:	Util-linux-2.22.2 -- description
mktemp:	Coreutils-8.21 -- description
mk_cmds:	E2fsprogs-1.42.7 -- description
mmroff:	Groff-1.22.2 -- description
modinfo:	Kmod-12 -- description
modprobe:	Kmod-12 -- description
more:	Util-linux-2.22.2 -- description
mount:	Util-linux-2.22.2 -- description
mountpoint:	Util-linux-2.22.2 -- description
msgattrib:	Gettext-0.18.2 -- description
msgcat:	Gettext-0.18.2 -- description
msgcmp:	Gettext-0.18.2 -- description
msgcomm:	Gettext-0.18.2 -- description
msgconv:	Gettext-0.18.2 -- description
msgen:	Gettext-0.18.2 -- description
msgexec:	Gettext-0.18.2 -- description
msgfilter:	Gettext-0.18.2 -- description
msgfmt:	Gettext-0.18.2 -- description
msggrep:	Gettext-0.18.2 -- description
msginit:	Gettext-0.18.2 -- description
msgmerge:	Gettext-0.18.2 -- description
msgunfmt:	Gettext-0.18.2 -- description
msguniq:	Gettext-0.18.2 -- description
mtrace:	Glibc-2.17 -- description
mv:	Coreutils-8.21 -- description
namei:	Util-linux-2.22.2 -- description
ncursesw5-config:	Ncurses-5.9 -- description
neqn:	Groff-1.22.2 -- description
newgrp:	Shadow-4.1.5.1 -- description
newusers:	Shadow-4.1.5.1 -- description
ngettext:	Gettext-0.18.2 -- description
nice:	Coreutils-8.21 -- description
nl:	Coreutils-8.21 -- description
nm:	Binutils-2.23.1 -- description
nohup:	Coreutils-8.21 -- description
nologin:	Shadow-4.1.5.1 -- description
nproc:	Coreutils-8.21 -- description
nroff:	Groff-1.22.2 -- description
nscd:	Glibc-2.17 -- description
nstat:	IPRoute2-3.8.0 -- description
objcopy:	Binutils-2.23.1 -- description
objdump:	Binutils-2.23.1 -- description
od:	Coreutils-8.21 -- description
oldfind:	Findutils-4.4.2 -- description
openvt:	Kbd-1.15.5 -- description
partx:	Util-linux-2.22.2 -- description
passwd:	Shadow-4.1.5.1 -- description
paste:	Coreutils-8.21 -- description
patch:	Patch-2.7.1 -- description
pathchk:	Coreutils-8.21 -- description
pcprofiledump:	Glibc-2.17 -- description
pdfroff:	Groff-1.22.2 -- description
pdftexi2dvi:	Texinfo-5.0 -- description
peekfd:	Psmisc-22.20 -- description
perl:	Perl-5.16.2 -- description

perl5.16.2:	Perl-5.16.2 -- description
perlbug:	Perl-5.16.2 -- description
perldoc:	Perl-5.16.2 -- description
perlivp:	Perl-5.16.2 -- description
perlthanks:	Perl-5.16.2 -- description
pfbtops:	Groff-1.22.2 -- description
pg:	Util-linux-2.22.2 -- description
pgawk:	Gawk-4.0.2 -- description
pgawk-4.0.2:	Gawk-4.0.2 -- description
pgrep:	Procps-ng-3.3.6 -- description
pic:	Groff-1.22.2 -- description
pic2graph:	Groff-1.22.2 -- description
piconv:	Perl-5.16.2 -- description
pidof:	Sysvinit-2.88dsf -- description
ping:	Inetutils-1.9.1 -- description
ping6:	Inetutils-1.9.1 -- description
pinky:	Coreutils-8.21 -- description
pivot_root:	Util-linux-2.22.2 -- description
pkg-config:	Pkg-config-0.28 -- description
pkill:	Procps-ng-3.3.6 -- description
pl2pm:	Perl-5.16.2 -- description
pldd:	Glibc-2.17 -- description
pmap:	Procps-ng-3.3.6 -- description
pod2html:	Perl-5.16.2 -- description
pod2latex:	Perl-5.16.2 -- description
pod2man:	Perl-5.16.2 -- description
pod2texi:	Texinfo-5.0 -- description
pod2text:	Perl-5.16.2 -- description
pod2usage:	Perl-5.16.2 -- description
podchecker:	Perl-5.16.2 -- description
podselect:	Perl-5.16.2 -- description
post-grohtml:	Groff-1.22.2 -- description
poweroff:	Sysvinit-2.88dsf -- description
pr:	Coreutils-8.21 -- description
pre-grohtml:	Groff-1.22.2 -- description
preconv:	Groff-1.22.2 -- description
printenv:	Coreutils-8.21 -- description
printf:	Coreutils-8.21 -- description
prlimit:	Util-linux-2.22.2 -- description
prove:	Perl-5.16.2 -- description
prtstat:	Psmisc-22.20 -- description
ps:	Procps-ng-3.3.6 -- description
psed:	Perl-5.16.2 -- description
psfaddtable:	Kbd-1.15.5 -- description
psfgettable:	Kbd-1.15.5 -- description
psfstriptable:	Kbd-1.15.5 -- description
psfxtable:	Kbd-1.15.5 -- description
pstree:	Psmisc-22.20 -- description
pstree.x11:	Psmisc-22.20 -- description
pstruct:	Perl-5.16.2 -- description
ptar:	Perl-5.16.2 -- description
ptardiff:	Perl-5.16.2 -- description
ptargrep:	Perl-5.16.2 -- description
ptx:	Coreutils-8.21 -- description
pt_chown:	Glibc-2.17 -- description
pwcat:	Gawk-4.0.2 -- description
pwck:	Shadow-4.1.5.1 -- description

pwconv:	Shadow-4.1.5.1 -- description
pwd:	Coreutils-8.21 -- description
pwdx:	Procps-ng-3.3.6 -- description
pwunconv:	Shadow-4.1.5.1 -- description
py-compile:	Automake-1.13.1 -- description
ranlib:	Binutils-2.23.1 -- description
raw:	Util-linux-2.22.2 -- description
rcp:	Inetutils-1.9.1 -- description
readelf:	Binutils-2.23.1 -- description
readlink:	Coreutils-8.21 -- description
readprofile:	Util-linux-2.22.2 -- description
realpath:	Coreutils-8.21 -- description
reboot:	Sysvinit-2.88dsf -- description
recode-sr-latin:	Gettext-0.18.2 -- description
refer:	Groff-1.22.2 -- description
rename:	Util-linux-2.22.2 -- description
renice:	Util-linux-2.22.2 -- description
reset:	Ncurses-5.9 -- description
resize2fs:	E2fsprogs-1.42.7 -- description
resizepart:	Util-linux-2.22.2 -- description
rev:	Util-linux-2.22.2 -- description
rexec:	Inetutils-1.9.1 -- description
rlogin:	Inetutils-1.9.1 -- description
rm:	Coreutils-8.21 -- description
rmdir:	Coreutils-8.21 -- description
rmmod:	Kmod-12 -- description
rmt:	Tar-1.26 -- description
roff2dvi:	Groff-1.22.2 -- description
roff2html:	Groff-1.22.2 -- description
roff2pdf:	Groff-1.22.2 -- description
roff2ps:	Groff-1.22.2 -- description
roff2text:	Groff-1.22.2 -- description
roff2x:	Groff-1.22.2 -- description
routef:	IPRoute2-3.8.0 -- description
routel:	IPRoute2-3.8.0 -- description
rpcgen:	Glibc-2.17 -- description
rsh:	Inetutils-1.9.1 -- description
rtacct:	IPRoute2-3.8.0 -- description
rtcwake:	Util-linux-2.22.2 -- description
rtmon:	IPRoute2-3.8.0 -- description
rtpr:	IPRoute2-3.8.0 -- description
rtstat:	IPRoute2-3.8.0 -- description
runcon:	Coreutils-8.21 -- description
runlevel:	Sysvinit-2.88dsf -- description
runtest:	DejaGNU-1.5 -- description
rview:	Vim-7.3 -- description
rvim:	Vim-7.3 -- description
s2p:	Perl-5.16.2 -- description
script:	Util-linux-2.22.2 -- description
scriptreplay:	Util-linux-2.22.2 -- description
scsi_id:	Udev-197 (Extracted from systemd-197) -- description
sdiff:	Diffutils-3.2 -- description
sed:	Sed-4.2.2 -- description
seq:	Coreutils-8.21 -- description
setarch:	Util-linux-2.22.2 -- description
setfont:	Kbd-1.15.5 -- description
setkeycodes:	Kbd-1.15.5 -- description

setleds:	Kbd-1.15.5 -- description
setmetamode:	Kbd-1.15.5 -- description
setsid:	Util-linux-2.22.2 -- description
setterm:	Util-linux-2.22.2 -- description
sfdisk:	Util-linux-2.22.2 -- description
sg:	Shadow-4.1.5.1 -- description
sh:	Bash-4.2 -- description
sha1sum:	Coreutils-8.21 -- description
sha224sum:	Coreutils-8.21 -- description
sha256sum:	Coreutils-8.21 -- description
sha384sum:	Coreutils-8.21 -- description
sha512sum:	Coreutils-8.21 -- description
shasum:	Perl-5.16.2 -- description
showconsolefont:	Kbd-1.15.5 -- description
showkey:	Kbd-1.15.5 -- description
shred:	Coreutils-8.21 -- description
shuf:	Coreutils-8.21 -- description
shutdown:	Sysvinit-2.88dsf -- description
size:	Binutils-2.23.1 -- description
slabtop:	Procps-ng-3.3.6 -- description
sleep:	Coreutils-8.21 -- description
sln:	Glibc-2.17 -- description
soelim:	Groff-1.22.2 -- description
sort:	Coreutils-8.21 -- description
sotruss:	Glibc-2.17 -- description
splain:	Perl-5.16.2 -- description
split:	Coreutils-8.21 -- description
sprof:	Glibc-2.17 -- description
ss:	IPRoute2-3.8.0 -- description
stat:	Coreutils-8.21 -- description
stdbuf:	Coreutils-8.21 -- description
strings:	Binutils-2.23.1 -- description
strip:	Binutils-2.23.1 -- description
stty:	Coreutils-8.21 -- description
su:	Shadow-4.1.5.1 -- description
sulogin:	Sysvinit-2.88dsf -- description
sum:	Coreutils-8.21 -- description
swaplabel:	Util-linux-2.22.2 -- description
swapoff:	Util-linux-2.22.2 -- description
swapon:	Util-linux-2.22.2 -- description
switch_root:	Util-linux-2.22.2 -- description
symlink-tree:	Automake-1.13.1 -- description
sync:	Coreutils-8.21 -- description
sysctl:	Procps-ng-3.3.6 -- description
syslogd:	Sysklogd-1.5 -- description
tabs:	Ncurses-5.9 -- description
tac:	Coreutils-8.21 -- description
tail:	Coreutils-8.21 -- description
tailf:	Util-linux-2.22.2 -- description
talk:	Inetutils-1.9.1 -- description
tar:	Tar-1.26 -- description
taskset:	Util-linux-2.22.2 -- description
tbl:	Groff-1.22.2 -- description
tc:	IPRoute2-3.8.0 -- description
tclsh:	Tcl-8.6.0 -- description
tclsh8.6:	Tcl-8.6.0 -- description
tee:	Coreutils-8.21 -- description

telinit:	Sysvinit-2.88dsf -- description
telnet:	Inetutils-1.9.1 -- description
test:	Coreutils-8.21 -- description
testgdbm:	GDBM-1.10 -- description
texi2dvi:	Texinfo-5.0 -- description
texi2pdf:	Texinfo-5.0 -- description
texi2any:	Texinfo-5.0 -- description
texindex:	Texinfo-5.0 -- description
tfmtodit:	Groff-1.22.2 -- description
tftp:	Inetutils-1.9.1 -- description
tic:	Ncurses-5.9 -- description
timeout:	Coreutils-8.21 -- description
tload:	Procps-ng-3.3.6 -- description
toe:	Ncurses-5.9 -- description
top:	Procps-ng-3.3.6 -- description
touch:	Coreutils-8.21 -- description
tput:	Ncurses-5.9 -- description
tr:	Coreutils-8.21 -- description
traceroute:	Inetutils-1.9.1 -- description
troff:	Groff-1.22.2 -- description
true:	Coreutils-8.21 -- description
truncate:	Coreutils-8.21 -- description
tset:	Ncurses-5.9 -- description
tsort:	Coreutils-8.21 -- description
tty:	Coreutils-8.21 -- description
tune2fs:	E2fsprogs-1.42.7 -- description
tunelp:	Util-linux-2.22.2 -- description
tzselect:	Glibc-2.17 -- description
udevadm:	Udev-197 (Extracted from systemd-197) -- description
udevd:	Udev-197 (Extracted from systemd-197) -- description
ul:	Util-linux-2.22.2 -- description
umount:	Util-linux-2.22.2 -- description
uname:	Coreutils-8.21 -- description
uncompress:	Gzip-1.5 -- description
unexpand:	Coreutils-8.21 -- description
unicode_start:	Kbd-1.15.5 -- description
unicode_stop:	Kbd-1.15.5 -- description
uniq:	Coreutils-8.21 -- description
unlink:	Coreutils-8.21 -- description
unlzma:	Xz-5.0.4 -- description
unshare:	Util-linux-2.22.2 -- description
unxz:	Xz-5.0.4 -- description
updatedb:	Findutils-4.4.2 -- description
uptime:	Procps-ng-3.3.6 -- description
useradd:	Shadow-4.1.5.1 -- description
userdel:	Shadow-4.1.5.1 -- description
usermod:	Shadow-4.1.5.1 -- description
users:	Coreutils-8.21 -- description
utmpdump:	Util-linux-2.22.2 -- description
uuidd:	Util-linux-2.22.2 -- description
uuidgen:	Util-linux-2.22.2 -- description
vdir:	Coreutils-8.21 -- description
vi:	Vim-7.3 -- description
view:	Vim-7.3 -- description
vigr:	Shadow-4.1.5.1 -- description
vim:	Vim-7.3 -- description
vimdiff:	Vim-7.3 -- description

vimtutor:	Vim-7.3 -- description
vipw:	Shadow-4.1.5.1 -- description
vmstat:	Procps-ng-3.3.6 -- description
w:	Procps-ng-3.3.6 -- description
wall:	Util-linux-2.22.2 -- description
watch:	Procps-ng-3.3.6 -- description
wc:	Coreutils-8.21 -- description
wdctl:	Util-linux-2.22.2 -- description
whatis:	Man-DB-2.6.3 -- description
whereis:	Util-linux-2.22.2 -- description
who:	Coreutils-8.21 -- description
whoami:	Coreutils-8.21 -- description
wipefs:	Util-linux-2.22.2 -- description
x86_64:	Util-linux-2.22.2 -- description
xargs:	Findutils-4.4.2 -- description
xgettext:	Gettext-0.18.2 -- description
xsubpp:	Perl-5.16.2 -- description
xtrace:	Glibc-2.17 -- description
xxd:	Vim-7.3 -- description
xz:	Xz-5.0.4 -- description
xzcat:	Xz-5.0.4 -- description
xzcmp:	Xz-5.0.4 -- description
xzdec:	Xz-5.0.4 -- description
xzdiff:	Xz-5.0.4 -- description
xzegrep:	Xz-5.0.4 -- description
xzfgrep:	Xz-5.0.4 -- description
xzgrep:	Xz-5.0.4 -- description
xzless:	Xz-5.0.4 -- description
xzmore:	Xz-5.0.4 -- description
yacc:	Bison-2.7 -- description
yes:	Coreutils-8.21 -- description
ylwrap:	Automake-1.13.1 -- description
zcat:	Gzip-1.5 -- description
zcmp:	Gzip-1.5 -- description
zdiff:	Gzip-1.5 -- description
zdump:	Glibc-2.17 -- description
zegrep:	Gzip-1.5 -- description
zfgrep:	Gzip-1.5 -- description
zforce:	Gzip-1.5 -- description
zgrep:	Gzip-1.5 -- description
zic:	Glibc-2.17 -- description
zipdetails:	Perl-5.16.2 -- description
zless:	Gzip-1.5 -- description
zmore:	Gzip-1.5 -- description
znew:	Gzip-1.5 -- description
zsoelim:	Man-DB-2.6.3 -- description

Libraries

ld.so:	Glibc-2.17 -- description
libanl:	Glibc-2.17 -- description
libasprintf:	Gettext-0.18.2 -- description
libbfd:	Binutils-2.23.1 -- description
libblkid:	Util-linux-2.22.2 -- description
libBrokenLocale:	Glibc-2.17 -- description
libbsd-compat:	Glibc-2.17 -- description
libbz2*:	Bzip2-1.0.6 -- description

libc:	Glibc-2.17 -- description
libcheck:	Check-0.9.9 -- description
libcidn:	Glibc-2.17 -- description
libcom_err:	E2fsprogs-1.42.7 -- description
libcrypt:	Glibc-2.17 -- description
libcurses:	Ncurses-5.9 -- description
libdl:	Glibc-2.17 -- description
libe2p:	E2fsprogs-1.42.7 -- description
libexpect-5.45:	Expect-5.45 -- description
libext2fs:	E2fsprogs-1.42.7 -- description
libfl.a:	Flex-2.5.37 -- description
libform:	Ncurses-5.9 -- description
libg:	Glibc-2.17 -- description
libgcc*:	GCC-4.7.2 -- description
libgcov:	GCC-4.7.2 -- description
libgdbm:	GDBM-1.10 -- description
libgettextlib:	Gettext-0.18.2 -- description
libgettextpo:	Gettext-0.18.2 -- description
libgettextsrc:	Gettext-0.18.2 -- description
libgmp:	GMP-5.1.1 -- description
libgmpxx:	GMP-5.1.1 -- description
libgomp:	GCC-4.7.2 -- description
libhistory:	Readline-6.2 -- description
libiberty:	Binutils-2.23.1 -- description
libieee:	Glibc-2.17 -- description
libkmod:	Kmod-12
libltdl:	Libtool-2.4.2 -- description
liblto_plugin*:	GCC-4.7.2 -- description
liblzma*:	Xz-5.0.4 -- description
libm:	Glibc-2.17 -- description
libmagic:	File-5.13 -- description
libman:	Man-DB-2.6.3 -- description
libmandb:	Man-DB-2.6.3 -- description
libmcheck:	Glibc-2.17 -- description
libmemusage:	Glibc-2.17 -- description
libmenu:	Ncurses-5.9 -- description
libmount:	Util-linux-2.22.2 -- description
libmp:	GMP-5.1.1 -- description
libmpc:	MPC-1.0.1 -- description
libmpfr:	MPFR-3.1.1 -- description
libmudflap*:	GCC-4.7.2 -- description
libncurses:	Ncurses-5.9 -- description
libnsl:	Glibc-2.17 -- description
libnss:	Glibc-2.17 -- description
libopcodes:	Binutils-2.23.1 -- description
libpanel:	Ncurses-5.9 -- description
libpcprofile:	Glibc-2.17 -- description
libpipeline:	Libpipeline-1.2.2
libprocps:	Procps-ng-3.3.6 -- description
libpthread:	Glibc-2.17 -- description
libquadmath*:	GCC-4.7.2 -- description
libquota:	E2fsprogs-1.42.7 -- description
libreadline:	Readline-6.2 -- description
libresolv:	Glibc-2.17 -- description
librpcsvc:	Glibc-2.17 -- description
librt:	Glibc-2.17 -- description
libSegFault:	Glibc-2.17 -- description

libss:	E2fsprogs-1.42.7 -- description
libssp*:	GCC-4.7.2 -- description
libstdbuf.so:	Coreutils-8.21 -- description
libstdc++:	GCC-4.7.2 -- description
libsupc++:	GCC-4.7.2 -- description
libtcl8.6.so:	Tcl-8.6.0 -- description
libtclstub8.6.a:	Tcl-8.6.0 -- description
libthread_db:	Glibc-2.17 -- description
libudev:	Udev-197 (Extracted from systemd-197) -- description
libutil:	Glibc-2.17 -- description
libuuid:	Util-linux-2.22.2 -- description
liby.a:	Bison-2.7 -- description
libz:	Zlib-1.2.7 -- description
preloadable_libintl:	Gettext-0.18.2 -- description

Scripts

checkfs:	LFS-Bootscripts-20130123 -- description
cleanfs:	LFS-Bootscripts-20130123 -- description
console:	LFS-Bootscripts-20130123 -- description
configuring:	Configuring the Linux Console
functions:	LFS-Bootscripts-20130123 -- description
halt:	LFS-Bootscripts-20130123 -- description
hostname: configuring:	Configuring the system hostname
ifdown:	LFS-Bootscripts-20130123 -- description
ifup:	LFS-Bootscripts-20130123 -- description
localnet:	LFS-Bootscripts-20130123 -- description
/etc/hosts:	Customizing the /etc/hosts File
modules:	LFS-Bootscripts-20130123 -- description
mountfs:	LFS-Bootscripts-20130123 -- description
mountkernfs:	LFS-Bootscripts-20130123 -- description
network:	LFS-Bootscripts-20130123 -- description
/etc/hosts:	Customizing the /etc/hosts File
configuring:	General Network Configuration
rc:	LFS-Bootscripts-20130123 -- description
reboot:	LFS-Bootscripts-20130123 -- description
sendsignals:	LFS-Bootscripts-20130123 -- description
setclock:	LFS-Bootscripts-20130123 -- description
configuring:	Configuring the setclock Script
static:	LFS-Bootscripts-20130123 -- description
swap:	LFS-Bootscripts-20130123 -- description
sysctl:	LFS-Bootscripts-20130123 -- description
sysklogd:	LFS-Bootscripts-20130123 -- description
configuring:	Configuring the sysklogd Script
template:	LFS-Bootscripts-20130123 -- description
udev:	LFS-Bootscripts-20130123 -- description
udev_retry:	LFS-Bootscripts-20130123 -- description

Others

/boot/config-3.8.1:	Linux-3.8.1 -- description
/boot/System.map-3.8.1:	Linux-3.8.1 -- description
/dev/*:	Preparing Virtual Kernel File Systems
/etc/fstab:	Creating the /etc/fstab File
/etc/group:	Creating Essential Files and Symlinks
/etc/hosts:	Customizing the /etc/hosts File
/etc/inittab:	Configuring Sysvinit