

Cross-Compiled Linux From Scratch

Version GIT-20170803-MIPS

Cross-Compiled Linux From Scratch: Version GIT-20170803-MIPS

Copyright © 2005–2017 Joe Ciccone, Jim Gifford & Ryan Oliver

Based on LFS, Copyright © 1999–2017 Gerard Beekmans

Copyright © 2005–2017, Joe Ciccone, Jim Gifford, & Ryan Oliver

All rights reserved.

This material may be distributed only subject to the terms and conditions set forth in the Open Publication License v1.0 or later (the latest version is presently available at <http://www.opencontent.org/openpub/>).

Linux® is a registered trademark of Linus Torvalds.

This book is based on the "Linux From Scratch" book, that was released under the following license:

Copyright © 1999–2017, Gerard Beekmans

All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions in any form must retain the above copyright notice, this list of conditions and the following disclaimer
- Neither the name of "Linux From Scratch" nor the names of its contributors may be used to endorse or promote products derived from this material without specific prior written permission
- Any material derived from Linux From Scratch must contain a reference to the "Linux From Scratch" project

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Table of Contents

Preface	ix
i. Foreword	ix
ii. Audience	ix
iii. Prerequisites	x
iv. Host System Requirements	xi
v. Typography	xii
vi. Structure	xiii
vii. Errata	xiv
I. Introduction	1
1. Introduction	2
1.1. Cross-LFS Acknowledgements	2
1.2. How to Build a CLFS System	3
1.3. Recommended Build Information	4
1.4. Master Changelog	4
1.5. Changelog for MIPS 32 Bit	10
1.6. Resources	10
1.7. Help	11
II. Preparing for the Build	13
2. Preparing a New Partition	14
2.1. Introduction	14
2.2. Setting The <code>\${CLFS}</code> Variable	14
2.3. Creating a New Partition	14
2.4. Creating a File System on the Partition	15
2.5. Mounting the New Partition	16
3. Packages and Patches	18
3.1. Introduction	18
3.2. All Packages	18
3.3. Additional Packages for MIPS	26
3.4. Needed Patches	26
3.5. Additional Patches for MIPS	27
4. Final Preparations	28
4.1. Introduction	28
4.2. Creating the <code>\${CLFS}/tools</code> Directory	28
4.3. Creating the <code>\${CLFS}/cross-tools</code> Directory	28
4.4. Adding the CLFS User	29
4.5. Setting Up the Environment	30
4.6. Build Variables	31
4.7. About the Test Suites	31
III. Make the Cross-Compile Tools	32
5. Constructing Cross-Compile Tools	33
5.1. Introduction	33
5.2. File-5.31	34
5.3. Linux-4.9.21 Headers	35
5.4. M4-1.4.18	36
5.5. Ncurses-6.0	37

5.6. Pkg-config-lite-0.28-1	38
5.7. GMP-6.1.2	39
5.8. MPFR-3.1.5	40
5.9. MPC-1.0.3	41
5.10. ISL-0.17.1	42
5.11. Cross Binutils-2.28	43
5.12. Cross GCC-7.1.0 - Static	45
5.13. Glibc-2.25	48
5.14. Cross GCC-7.1.0 - Final	50
IV. Building the Basic Tools	52
6. Constructing a Temporary System	53
6.1. Introduction	53
6.2. Build Variables	53
6.3. GMP-6.1.2	54
6.4. MPFR-3.1.5	55
6.5. MPC-1.0.3	56
6.6. ISL-0.17.1	57
6.7. Zlib-1.2.11	58
6.8. Binutils-2.28	59
6.9. GCC-7.1.0	60
6.10. Ncurses-6.0	62
6.11. Bash-4.4	63
6.12. Bzip2-1.0.6	65
6.13. Check-0.11.0	66
6.14. Coreutils-8.27	67
6.15. Diffutils-3.6	68
6.16. File-5.31	69
6.17. Findutils-4.6.0	70
6.18. Gawk-4.1.4	71
6.19. Gettext-0.19.8.1	72
6.20. Grep-3.0	73
6.21. Gzip-1.8	74
6.22. Make-4.2.1	75
6.23. Patch-2.7.5	76
6.24. Sed-4.4	77
6.25. Tar-1.29	78
6.26. Texinfo-6.3	79
6.27. Util-linux-2.29.2	80
6.28. Vim-8.0	81
6.29. XZ Utils-5.2.3	83
6.30. To Boot or to Chroot?	84
7. If You Are Going to Boot	85
7.1. Introduction	85
7.2. Bc-1.07.1	86
7.3. Boot-scripts for CLFS 3.0-20140710	87
7.4. E2fsprogs-1.43.4	89
7.5. Kmod-24	90

7.6. Shadow-4.5	91
7.7. Sysvinit-2.88dsf	92
7.8. Eudev-1.7	95
7.9. Linux-4.9.21	97
7.10. Bootloaders	99
7.11. Colo-1.22	100
7.12. Creating Directories	101
7.13. Creating Essential Symlinks	101
7.14. Populating /dev	103
7.15. Creating the passwd and group Files	103
7.16. Creating the /etc/fstab File	105
7.17. Setting Up the Environment	105
7.18. Changing Ownership	106
7.19. How to View the Book	106
7.20. Making the Temporary System Bootable	106
7.21. What to do next	107
8. If You Are Going to Chroot	108
8.1. Introduction	108
8.2. Mounting Virtual Kernel File Systems	108
8.3. Before Entering the Chroot Environment	109
8.4. Entering the Chroot Environment	110
8.5. Changing Ownership	111
8.6. Creating Directories	111
8.7. Creating Essential Symlinks	112
8.8. Creating the passwd and group Files	113
V. Building the CLFS System	116
9. Constructing Testsuite Tools	117
9.1. Introduction	117
9.2. Tcl-8.6.4	118
9.3. Expect-5.45	119
9.4. DejaGNU-1.6	120
10. Installing Basic System Software	121
10.1. Introduction	121
10.2. Package Management	121
10.3. About Test Suites, Again	124
10.4. Temporary Perl-5.26.0	125
10.5. Linux-4.9.21 Headers	126
10.6. Man-pages-4.09	127
10.7. Glibc-2.25	128
10.8. Adjusting the Toolchain	135
10.9. M4-1.4.18	136
10.10. GMP-6.1.2	137
10.11. MPFR-3.1.5	139
10.12. MPC-1.0.3	140
10.13. ISL-0.17.1	141
10.14. Zlib-1.2.11	142
10.15. Flex-2.6.4	143

10.16. Bison-3.0.4	144
10.17. Binutils-2.28	145
10.18. GCC-7.1.0	148
10.19. Attr-2.4.47	151
10.20. Acl-2.2.52	152
10.21. Libcap-2.25	154
10.22. Sed-4.4	155
10.23. Pkg-config-lite-0.28-1	156
10.24. Ncurses-6.0	157
10.25. Shadow-4.5	159
10.26. Util-linux-2.29.2 Pass 1	162
10.27. Procps-ng-3.3.12	163
10.28. E2fsprogs-1.43.4	165
10.29. Coreutils-8.27	168
10.30. Iana-Etc-2.30	173
10.31. Libtool-2.4.6	174
10.32. IPRoute2-4.9.0	175
10.33. Bzip2-1.0.6	177
10.34. GDBM-1.13	179
10.35. Perl-5.26.0	180
10.36. Readline-7.0	183
10.37. Autoconf-2.69	184
10.38. Automake-1.15	185
10.39. Bash-4.4	187
10.40. Bc-1.07.1	189
10.41. Diffutils-3.6	190
10.42. File-5.31	191
10.43. Gawk-4.1.4	192
10.44. Findutils-4.6.0	193
10.45. Gettext-0.19.8.1	194
10.46. Gperf-3.0.4	196
10.47. Grep-3.0	197
10.48. Groff-1.22.3	198
10.49. Less-491	201
10.50. Gzip-1.8	202
10.51. IPutils-s20150815	204
10.52. Kbd-2.0.4	205
10.53. Libpipeline-1.4.1	207
10.54. Man-DB-2.7.6.1	208
10.55. Make-4.2.1	211
10.56. XZ Utils-5.2.3	212
10.57. Expat-2.2.0	214
10.58. XML::Parser-2.44	215
10.59. Intltool-0.51.0	216
10.60. Kmod-24	217
10.61. Patch-2.7.5	219
10.62. Psmisc-22.21	220

10.63. Systemd-233	221
10.64. D-Bus-1.10.18	226
10.65. Tar-1.29	228
10.66. Texinfo-6.3	229
10.67. Util-linux-2.29.2	230
10.68. Vim-8.0	235
10.69. Colo-1.22	238
10.70. Dvhtool-1.0.1	240
10.71. Arcload-0.5	241
10.72. About Debugging Symbols	242
10.73. Stripping	242
11. System Configuration	243
11.1. Introduction	243
11.2. How does Systemd work?	243
11.3. Configuring the system clock	244
11.4. Configuring the Linux Console	245
11.5. Device and Module Handling on a CLFS System	246
11.6. Creating custom symlinks to devices	249
11.7. The Bash Shell Startup Files	251
11.8. Setting Up Locale Information	252
11.9. Creating the /etc/inputrc File	253
11.10. Creating the /etc/fstab File	255
12. Networking Configuration	256
12.1. Configuring the system hostname	256
12.2. Customizing the /etc/hosts File	256
12.3. Creating the /etc/resolv.conf File	257
12.4. Systemd Networking?	257
12.5. Networking Configuration with Systemd-networkd	258
12.6. CLFS-Network-Scripts-20140224	260
12.7. Static Networking Configuration	261
12.8. DHCPD-6.11.5	263
13. Making the CLFS System Bootable	265
13.1. Introduction	265
13.2. Linux-4.9.21	266
13.3. Making the CLFS System Bootable via Arcload	270
13.4. Making the CLFS System Bootable via Colo	270
14. The End	272
14.1. The End	272
14.2. Download Client	272
14.3. Rebooting the System	273
14.4. What Now?	274
VI. Appendices	276
A. Acronyms and Terms	277
B. Dependencies	280
C. Mips Dependencies	290
D. Package Rationale	291
E. Package Rationale - MIPS	296

F. Open Publication License	297
Index	300

Preface

Foreword

The Linux From Scratch Project has seen many changes in the few years of its existence. I personally became involved with the project in 1999, around the time of the 2.x releases. At that time, the build process was to create static binaries with the host system, then chroot and build the final binaries on top of the static ones.

Later came the use of the /static directory to hold the initial static builds, keeping them separated from the final system, then the PureLFS process developed by Ryan Oliver and Greg Schafer, introducing a new toolchain build process that divorces even our initial builds from the host. Finally, LFS 6 brought Linux Kernel 2.6, the udev dynamic device structure, sanitized kernel headers, and other improvements to the Linux From Scratch system.

The one "flaw" in LFS is that it has always been based on an x86 class processor. With the advent of the Athlon 64 and Intel EM64T processors, the x86-only LFS is no longer ideal. Throughout this time, Ryan Oliver developed and documented a process by which you could build Linux for any system and from any system, by use of cross-compilation techniques. Thus, the Cross-Compiled LFS (CLFS) was born.

CLFS follows the same guiding principles the LFS project has always followed, e.g., knowing your system inside and out by virtue of having built the system yourself. Additionally, during a CLFS build, you will learn advanced techniques such as cross-build toolchains, multilib support (32 & 64-bit libraries side-by-side), alternative architectures such as Sparc, MIPS, and much more.

We hope you enjoy building your own CLFS system, and the benefits that come from a system tailored to your needs.

```
--
Jeremy Utle, CLFS 1.x Release Manager (Page Author)
Jonathan Norman, Release Manager
Jim Gifford, CLFS Project Co-leader
Ryan Oliver, CLFS Project Co-leader
Joe Ciccone, CLFS Project Co-leader
Jonathan Norman, Justin Knierim, Chris Staub, Matt Darcy, Ken Moffat,
Manuel Canales Esparcia, Nathan Coulson and William Harrington - CLFS Developers
```

Audience

There are many reasons why somebody would want to read this book. The principal reason is to install a Linux system from the source code. A question 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?" That is a good question and is the impetus for this section of the book.

One important reason for the existence of CLFS is to help people understand how a Linux system works. Building an CLFS system helps demonstrate what makes Linux tick, and how things work together and depend on each other. One of the best things this learning experience provides is the ability to customize Linux to your own tastes and needs.

A key benefit of CLFS is that it allows users to have more control over their system without any reliance on a Linux implementation designed by someone else. With CLFS, *you* are in the driver's seat and dictate every aspect of the system, such as the directory layout and bootscript setup. You also dictate where, why, and how programs are installed.

Another benefit of CLFS is the ability to create a very compact Linux system. When installing a regular distribution, one is often forced to include several programs which are probably never used. These programs waste disk space or CPU cycles. It is not difficult to build an CLFS system of less than 100 megabytes (MB), which is substantially smaller than the majority of existing installations. Does this still sound like a lot of space? A few of us have been working on creating a very small embedded CLFS system. We successfully built a system that was specialized to run the Apache web server with approximately 8MB of disk space used. Further stripping could bring this down to 5 MB or less. Try that with a regular distribution! This is only one of the many benefits of designing your own Linux implementation.

We could compare Linux distributions to a hamburger purchased at a fast-food restaurant—you have no idea what might be in what you are eating. CLFS, on the other hand, does not give you a hamburger. Rather, CLFS provides the recipe to make the exact hamburger desired. This allows users to review the recipe, omit unwanted ingredients, and add your own ingredients to enhance the flavor of the burger. When you are satisfied with the recipe, move on to preparing it. It can be made to exact specifications—broil it, bake it, deep-fry it, or barbecue it.

Another analogy that we can use is that of comparing CLFS with a finished house. CLFS provides the skeletal plan of a house, but it is up to you to build it. CLFS maintains the freedom to adjust plans throughout the process, customizing it to the needs and preferences of the user.

Security is an additional advantage of a custom built Linux system. 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 Cross Linux From Scratch is to build a complete and usable foundation-level system. Readers who do not wish to build their own Linux system from scratch may not benefit from the information in this book. If you only want to know what happens while the computer boots, we recommend the “From Power Up To Bash Prompt” HOWTO located at <http://axiom.anu.edu.au/~okeefe/p2b/> or on The Linux Documentation Project's (TLDP) website at <http://www.tldp.org/HOWTO/From-PowerUp-To-Bash-Prompt-HOWTO.html>. The HOWTO builds a system which is similar to that of this book, but it focuses strictly on creating a system capable of booting to a BASH prompt. Consider your objective. If you wish to build a Linux system and learn along the way, this book is your best choice.

There are too many good reasons to build your own CLFS system to list them all here. This section is only the tip of the iceberg. As you continue in your CLFS experience, you will find the power that information and knowledge truly bring.

Prerequisites

Building a CLFS 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, the reader 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 the reader has a reasonable knowledge of using and installing Linux software. A basic knowledge of the architectures being used in the Cross LFS process and the host operating systems in use is also required.

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

Before building a CLFS 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 distributions under Linux.

- The Linux Users' Guide

<http://www.tldp.org/pub/Linux/docs/ldp-archived/users-guide/>

This guide covers the usage of assorted Linux software.

- The Essential Pre-Reading Hint

http://hints.clfs.org/index.php/Essential_Prereading

This is a 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 CLFS should have an understanding of many of the topics in this hint.

Host System Requirements

You should be able to build a CLFS system from just about any Unix-type operating system. Your host system should have the following software with the minimum versions indicated. 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.

- **Bash-2.05a**
- **Binutils-2.12** (Versions greater than 2.28 are not recommended as they have not been tested)
- **Bison-1.875**
- **Bzip2-1.0.2**
- **Coreutils-5.0**
- **Diffutils-2.8**
- **Findutils-4.1.20**
- **Gawk-3.1.5**
- **GCC-4.1.2** and the C++ compiler, **g++** (Versions greater than 7.1.0 are not recommended as they have not been tested)
- **Glibc-2.2.5** (Versions greater than 2.25 are not recommended as they have not been tested)
- **Grep-2.5**
- **Gzip-1.2.4**
- **Make-3.80**
- **Ncurses-5.3**
- **Patch-2.5.4**
- **Sed-3.0.2**
- **Tar-1.22**
- **Texinfo-4.7**
- **XZ Utils-4.999.8beta**

To see whether your host system has all the appropriate versions, create and run the following script. Read the output carefully for any errors, and make sure to install any packages that are reported as not found.

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

# Simple script to list version numbers of critical development tools

bash --version | head -n1 | cut -d" " -f2-4
echo -n "Binutils: "; ld --version | head -n1 | cut -d" " -f3-
bison --version | head -n1
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
gcc --version | head -n1
g++ --version | head -n1
ldd $(which ${SHELL}) | grep libc.so | cut -d ' ' -f 3 | ${SHELL} | head -n 1 | c
grep --version | head -n1
gzip --version | head -n1
make --version | head -n1
tic -V
patch --version | head -n1
sed --version | head -n1
tar --version | head -n1
makeinfo --version | head -n1
xz --version | head -n1
echo 'int main(){}' | gcc -v -o /dev/null -x c - > dummy.log 2>&1
if ! grep -q 'error' dummy.log; then
    echo "Compilation successful" && rm dummy.log
else
    echo 1>&2 "Compilation FAILED - more development packages may need to be \
installed. If you like, you can also view dummy.log for more details."
fi
EOF

bash version-check.sh 2>errors.log &&
[ -s errors.log ] && echo -e "\nThe following packages could not be found:\n$(cat
```

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 Cross-Compiled 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.

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

This form of text (fixed-width text) shows screen output, probably 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://clfs.org/>

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

```
cat > ${CLFS}/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 `${CLFS}/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 copied-and-pasted.

```
passwd(5)
```

This format is used to refer to a specific manual page (hereinafter referred to simply as a “man” page). The number inside parentheses indicates a specific section inside of **man**. For example, **passwd** has two man pages. Per CLFS installation instructions, those two man pages will be located at `/usr/share/man/man1/passwd.1` and `/usr/share/man/man5/passwd.5`. Both man pages have different information in them. 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 Cross-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 and downloading the packages.

Part III - Make the Cross-Compile Tools

Part III shows you how to make a set of Cross-Compiler tools. These tools can run on your host system but allow you to build packages that will run on your target system.

Part IV - Building the Basic Tools

Part IV explains how to build a tool chain designed to operate on your target system. These are the tools that will allow you to build a working system on your target computer.

Part V - Building the CLFS System

Part V guides the reader through the building of the CLFS 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.

Appendices

The appendices contain information that doesn't really fit anywhere else in the book. Appendix A contains definitions of acronyms and terms used in the book; Appendices B and C have information about package dependencies and the build order. Some architectures may have additional appendices for arch-specific issues.

Errata

The software used to create a CLFS system is constantly being updated and enhanced. Security warnings and bug fixes may become available after the CLFS book has been released. Some host systems may also have problems building CLFS. To check whether the package versions or instructions in this release of CLFS need any modifications to accommodate security vulnerabilities, other bug fixes, or host-specific issues, please visit <http://trac.clfs.org/wiki/errata> 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 CLFS system.

Part I. Introduction

Chapter 1. Introduction

1.1. Cross-LFS Acknowledgements

The CLFS team would like to acknowledge people who have assisted in making the book what it is today.

Our Leaders:

- William Harrington - Lead Developer.
- Jonathan Norman - x86, x86_64, PowerPC & UltraSPARC builds, Release Manager 2.x Series
- Chris Staub - x86 and x86_64 builds. Leader of Quality Control.

Our CLFS Team:

- Matt Darcy - x86, X86_64, and Sparc builds.
- Manuel Canales Esparcia - Book XML.
- Justin Knierim - Website Architect.
- Ken Moffat - PowerPC and X86_64 builds. Developer of Pure 64 Hint.

Outside the Development Team

- Jürg Billeter - Testing and assisting in the development of the Linux Headers Package
- Richard Downing - Testing, typo, and content fixes.
- Peter Ennis - Typo and content fixes.
- Tony Morgan - Typo and content fixes.

The CLFS team would also like to acknowledge contributions of people from *clfs-dev@lists.clfs.org* and associated mailing lists who have provided valuable technical and editorial corrections while testing the Cross-LFS book.

- G. Moko - Text updates and Typos
- Maxim Osipov - MIPS Testing.
- Doug Ronne - Various x86_64 fixes.
- Theo Schneider - Testing of the Linux Headers Package
- Martin Ward - Recommendations for Systemd and the Boot method, among other contributions
- William Zhou - Text updates and Typos

Former Team Members

- Joe Ciccone - Lead Developer.
- Nathan Coulson - Bootscripts.
- Jim Gifford - Lead Developer.
- Jeremy Huntwork - PowerPC, x86, Sparc builds.
- Karen McGuiness - Proofreader.
- Ryan Oliver - Build Process Developer.

- Alexander E. Patrakov - Udev/Hotplug Integration
- Jeremy Uteley - Release Manager 1.x Series.
- Zack Winkles - Unstable book work.

The Linux From Scratch Project

- *Gerard Beekmans* <gerard@linuxfromscratch.org> – Creator of Linux From Scratch, on which Cross-LFS is based

Thank you all for your support.

1.2. How to Build a CLFS System

The CLFS system will be built by using a previously installed Unix system or Linux distribution (such as Debian, Fedora, openSUSE, or Ubuntu). This existing 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 an entire separate distribution onto your machine, you may wish to use a livecd. Most distributions provide a livecd, which provides an environment to which you can add the required tools onto, allowing you to successfully follow the instructions in this book. Remember that if you reboot the livecd you will need to reconfigure the host environment before continuing with your build.

Preparing a New Partition of this book describes how to create a new Linux native partition and file system, the place where the new CLFS system will be compiled and installed. Packages and Patches explains which packages and patches need to be downloaded to build a CLFS system and how to store them on the new file system. Final Preparations discusses the setup for an appropriate working environment. Please read Final Preparations carefully as it explains several important issues the developer should be aware of before beginning to work through Constructing Cross-Compile Tools and beyond.

Constructing Cross-Compile Tools explains the installation of cross-compile tools which will be built on the host but be able to compile programs that run on the target machine. These cross-compile tools will be used to create a temporary, minimal system that will be the basis for building the final CLFS system. Some of these packages are needed to resolve circular dependencies—for example, to compile a compiler, you need a compiler.

The process of building cross-compile tools first involves building and installing all the necessary tools to create a build system for the target machine. With these cross-compiled tools, we eliminate any dependencies on the toolchain from our host distro.

After we build our “Cross-Tools”, we start building a very minimal working system in `/tools`, using the cross-toolchain in `/cross-tools`. Once the temporary system is finished, we perform a few additional tasks to prepare to enter this temporary build environment, either by booting or chrooting into it. For more details about the difference between these methods, see Section 6.30, “To Boot or to Chroot?”.

In Installing Basic System Software, after having booted or chrooted into the temporary build environment, the full CLFS system is built.

To finish the installation, several configuration files are created in System Configuration, and the kernel and boot loader are set up in Making the CLFS System Bootable. The End contains information on furthering the CLFS experience beyond this book. After the steps in this book have been implemented, the computer will be ready to reboot into the new CLFS 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 the reader embarks on the CLFS adventure.

1.3. Recommended Build Information

On the RaQ2, we are recommending the following:

The RaQ2 uses DOS style partitions, so build on a x86 and put the RaQ2 hard drive into that system.

Follow the directions using the reboot section.

Remove the hard drive and put it into the RaQ2 and continue your build.

On other MIPS based systems, you will have to build on the machine itself, since most of the other MIPS machines use SGI style partitions.

1.4. Master Changelog

This is version GIT-20170803 of the Cross-Compiled Linux From Scratch book, dated August 03, 2017. 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://trac.clfs.org/>.

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

Changelog Entries:

- 02 July 2017
 - [William Harrington] - Upgrade Diffutils to 3.6. Fixes ticket #1175.
- 09 June 2017
 - [William Harrington] - Upgrade PERL to 5.26.0. Fixes ticket #1167.
- 04 June 2017
 - [Chris] - Simplified sed command in temp-system Bzip2.
- 02 June 2017
 - [William Harrington] - Upgrade Sed to 4.4. Fixes ticket #1149.
- 01 June 2017
 - [William Harrington] - Upgrade Shadow to 4.5. Fixes ticket #1150.
- 29 May 2017
 - [William Harrington] - Upgrade File to 5.31. Fixes ticket #1163.
 - [William Harrington] - Upgrade Flex to 2.6.4. Fixes ticket #1089.
- 24 May 2017
 - [William Harrington] - Upgrade Coreutils to 8.27. Fixes ticket #1142.
- 19 May 2017
 - [William Harrington] - Upgrade Libcap to 2.25. Fixes ticket #1140.

- [William Harrington] - Move Libcap after Acl.
- 17 May 2017
 - [William Harrington] - Upgrade Vim to 8.0-0597. Fixes ticket #1154.
- 14 May 2017
 - [William Harrington] - Upgrade Perl to 5.24.1. Fixes ticket #1035.
 - [William Harrington] - Upgrade Binutils to 2.28. Fixes ticket #1108.
 - [William Harrington] - Upgrade ISL to 0.17.1. Fixes ticket #1114.
- 13 May 2017
 - [William Harrington] - Upgrade GCC to 7.1.0. Fixes ticket #1114.
- 10 May 2017
 - [William Harrington] - Upgrade MPFR to 3.1.5. Fixes ticket #1145.
- 06 May 2017
 - [William Harrington] - Upgrade GLIBC to 2.25. Fixes ticket #1109.
 - [William Harrington] - Upgrade Expat to 2.2.0. Fixes ticket #1130.
 - [William Harrington] - Upgrade Check to 0.11.0. Fixes ticket #1127.
- 04 May 2017
 - [William Harrington] - Upgrade Man-DB to 2.7.6.1. Fixes ticket #1143.
 - [William Harrington] - Upgrade Tar to 1.29. Fixes ticket #1151.
 - [William Harrington] - Upgrade BC to 1.07.1. Fixes ticket #1124.
 - [William Harrington] - Upgrade Grep to 3.0. Fixes ticket #1107.
 - [William Harrington] - Upgrade Readline to 7.0. Fixes ticket #1086.
- 03 May 2017
 - [William Harrington] - Upgrade Bash to 4.4. Fixes ticket #1085.
- 16 April 2017
 - [William Harrington] - Upgrade Util-linux to 2.29.2. Fixes ticket #1038.
 - [William Harrington] - Upgrade Systemd 233. Fixes ticket #1003.
 - [William Harrington] - Update group and user ids.
- 16 April 2017
 - [William Harrington] - Upgrade Linux kernel to 4.9.21. Fixes ticket #1105.
 - [William Harrington] - Upgrade IProute2 to 4.9.0. Fixes ticket #1103.
- 15 April 2017
 - [William Harrington] - Upgrade D-Bus to 1.10.18. Fixes ticket #1049.
 - [William Harrington] - Upgrade DejaGNU to 1.6. Fixes ticket #1128.
 - [William Harrington] - Upgrade DHCPD to 6.11.5. Fixes ticket #1077.
 - [William Harrington] - Upgrade Diffutils to 3.5. Fixes ticket #1129.

- [William Harrington] - Upgrade E2fsprogs to 1.43.4. Fixes ticket #1113.
- [William Harrington] - Upgrade File to 5.30. Fixes ticket #1131.
- [William Harrington] - Upgrade Findutils to 4.6.0. Fixes ticket #1101.
- [William Harrington] - Upgrade Gawk to 4.1.4. Fixes ticket #1132.
- [William Harrington] - Upgrade GDBM to 1.13. Fixes ticket #1133.
- [William Harrington] - Upgrade Gettext to 0.19.8.1. Fixes ticket #1134.
- [William Harrington] - Upgrade GMP to 6.1.2. Fixes ticket #1100.
- [William Harrington] - Upgrade GZIP to 1.8. Fixes ticket #1136.
- [William Harrington] - Upgrade KBD to 2.0.4. Fixes ticket #1137.
- [William Harrington] - Upgrade LESS to 491. Fixes ticket #1139.
- [William Harrington] - Upgrade M4 to 1.4.18. Fixes ticket #1142.
- [William Harrington] - Upgrade Make to 4.2.1. Fixes ticket #1112.
- [William Harrington] - Upgrade Man-pages to 4.09. Fixes ticket #1144.
- [William Harrington] - Upgrade Procs-ng to 3.3.12. Fixes ticket #1148.
- [William Harrington] - Upgrade Texinfo to 6.3. Fixes ticket #1152.
- [William Harrington] - Upgrade TZData to 2017b. Fixes ticket #1153.
- [William Harrington] - Upgrade XZ-Utils to 5.2.3. Fixes ticket #1157.
- [William Harrington] - Upgrade Zlib to 1.2.11. Fixes ticket #1156.
- 28 December 2016
 - [Chris] - Added "ext_attr" to the list of filesystem attributes to check for when creating a filesystem. Thanks to Roger Koehler for pointing this out.
- 13 March 2016
 - [Chris] - Updated Gawk dependencies - Gawk can use GMP, MPFR, Readline.
- 04 March 2016
 - [Chris] - Added creation of /var/log/faillog to Shadow instructions.
- 19 February 2016
 - [Chris] - Removed --disable-profile from Glibc instructions - profiling is disabled by default.
 - [Chris] - Removed commands to create libcurses* symlinks in Ncurses, as few packages now are likely to need them. Fixes ticket #1102.
 - [Chris] - Modified Ncurses instructions to move fewer libs to /lib{,32,64} as only libncursesw itself is needed there. Fixes ticket #1104.
 - [Chris] - Prevented the **bashbug** script from being installed in /bin, as it is not needed there. Fixes ticket #1099.
- 17 January 2016
 - [Chris] - Removed redundant commands from multilib Ncurses pages.
- 07 January 2016

- [Chris] - Removed command in Grep instructions for security fix that was addressed in latest version.
- 28 December 2015
 - [William Harrington] - Upgrade to GREP 2.22. Fixes ticket #1087.
 - [William Harrington] - Upgrade to LESS 481. Fixes ticket #1088.
 - [William Harrington] - Upgrade to Ncurses 6.0. Fixes ticket #1044.
 - [William Harrington] - Upgrade to GCC 5.3.0. Fixes ticket #1094.
- 23 December 2015
 - [Chris] - Changed LSB link to its new location at linuxfoundation.org.
- 19 December 2015
 - [Chris] - Added -j1 to vim install command to prevent errors in make install output.
- 13 December 2015
 - [Chris] - Edited configure options to have one per line. Fixes ticket #1091.
- 05 December 2015
 - [Chris] - Removed unneeded directory and symlink creation on Pure64 Essential Symlinks page.
- 30 November 2015
 - [Chris] - Removed one more remaining --disable-libstdcxx-pch option from final-system GCC installation.
 - [Chris] - Updates to command explanations.
- 27 November 2015
 - [William Harrington] - Update GCC to 5.2.0. Fixes ticket #1019.
 - [William Harrington] - Remove CLoog. Fixes ticket #1020.
 - [William Harrington] - Update ISL to 0.15. Fixes ticket #978.
- 17 October 2015
 - [William Harrington] - Update GLIBC to 2.22. Fixes ticket #1027.
- 16 October 2015
 - [William Harrington] - Update Binutils to 2.25.1. Fixes ticket #1023.
 - [William Harrington] - Update XZ to 5.2.2. Fixes ticket #1081.
 - [William Harrington] - Update KBD to 2.0.3. Fixes ticket #1057.
 - [William Harrington] - Update TZDATA to 2015g. Fixes ticket #1051.
 - [William Harrington] - Update Libpipeline to 1.4.1. Fixes ticket #1067.
- 01 October 2015
 - [Chris] - Removed unneeded --disable-static option from Cross-Tools File page, as that is now the default.
 - [Chris] - Removed --without-shared option from Cross-Tools Ncurses page, as that is the default.
- 22 September 2015
 - [William Harrington] - Update Libtool to 2.4.6. Fixes ticket #1076.
 - [William Harrington] - Update Gettext to 0.19.6. Fixes ticket #1075.

- [William Harrington] - Update File to 5.25. Fixes ticket #1074.
- 18 September 2015
 - [William Harrington] - Remove sed for E2fsprogs. Fixes ticket #1073.
- 16 September 2015
 - [William Harrington] - Update IProute2 to 4.1.1. Fixes ticket #992.
 - [William Harrington] - Update Linux to 4.1.7. Fixes ticket #995.
- 15 September 2015
 - [William Harrington] - Update Procps-ng to 3.3.11. Fixes ticket #1072.
- 07 September 2015
 - [William Harrington] - mv **gzexe** and **uncompress** to /usr/bin.
- 06 September 2015
 - [William Harrington] - Add IPv6 entry to /etc/hosts for localhost
 - [William Harrington] - Update iputils to s20150815. Fixes ticket #1066.
 - [William Harrington] - mv **fuser killall** to /bin per the FHS. Fixes ticket #1068.
 - [William Harrington] - Update MPFR to 3.1.3. Fixes ticket #1043.
 - [William Harrington] - Update Texinfo to 6.0. Fixes ticket #1041.
 - [William Harrington] - Update XML-Parser to 2.44. Fixes ticket #1039.
 - [William Harrington] - Update XZ to 5.2.1. Fixes ticket #1037.
 - [William Harrington] - Update Patch to 2.7.5. Fixes ticket #1034.
 - [William Harrington] - Update MPC to 1.0.3. Fixes ticket #1033.
 - [William Harrington] - Update KMOD to 21. Fixes ticket #1031.
 - [William Harrington] - Update Intltool to 0.51.0. Fixes ticket #1030.
 - [William Harrington] - Update Groff to 1.22.3. Fixes ticket #1029.
 - [William Harrington] - Update Grep to 2.21. Fixes ticket #1028.
 - [William Harrington] - Update DejaGNU to 1.5.3. Fixes ticket #1025.
 - [William Harrington] - Update Bison to 3.0.4. Fixes ticket #1024.
 - [William Harrington] - Update Automake to 1.15. Fixes ticket #1022.
 - [William Harrington] - Update LESS to 479. Fixes ticket #1056.
- 04 September 2015
 - [William Harrington] - Update File to 5.24. Fixes ticket #1021.
 - [William Harrington] - Update Man-db to 2.7.2. Fixes ticket #1032.
 - [William Harrington] - Update TCL to 8.6.4. Fixes ticket #1036.
 - [William Harrington] - Update Check to 0.10.0. Fixes ticket #1045.
 - [William Harrington] - Update E2fsprogs to 1.42.13. Fixes ticket #1046.
 - [William Harrington] - Update Gawk to 4.1.3. Fixes ticket #1047.
 - [William Harrington] - Update Gettext to 0.19.5.1. Fixes ticket #1048.

- [William Harrington] - Update Man-pages to 4.02. Fixes ticket #1050.
- [William Harrington] - Add Bc memory leak patch. Fixes ticket #1055.
- 30 July 2015
 - [Chris] - Removed reference to Freecode, as it's no longer being updated.
- 07 February 2015
 - [Chris] - Updated FHS URL to new location.
- 2 November 2014
 - [Chris] - Removed obsolete command for fixing **tzselect** from Glibc instructions.
 - [William Harrington] - Update E2FSprogs to 1.42.12. Fixes ticket #1012.
 - [William Harrington] - Update Libtool to 2.4.3. Fixes ticket #1013.
- 31 October 2014
 - [William Harrington] - Update TZDATA to 2014i. Fixes ticket #1006.
 - [William Harrington] - Update Util-linux to 2.25.2. Fixes ticket #1007.
 - [William Harrington] - Update XZ Utils to 5.0.7. Fixes ticket #1008.
 - [William Harrington] - Update DHCPCD to 6.6.0. Fixes ticket #1009.
 - [William Harrington] - Update Libpipeline to 1.4.0. Fixes ticket #1010.
 - [William Harrington] - Update D-Bus to 1.9.0. Fixes ticket #987.
 - [William Harrington] - Update GLIBC to 2.20. Fixes ticket #982.
 - [William Harrington] - Update Coreutils to 8.23. Fixes ticket #975.
- 25 October 2014
 - [William Harrington] - Update GREP to 2.20. Fixes ticket #969.
 - [William Harrington] - Update Check to 0.9.14. Fixes ticket #986.
 - [William Harrington] - Update DHCPCD to 6.5.1. Fixes ticket #988.
 - [William Harrington] - Update File to 5.20. Fixes ticket #990.
 - [William Harrington] - Update Gettext to 0.19.3. Fixes ticket #991.
 - [William Harrington] - Update KBD to 2.0.2. Fixes ticket #993.
 - [William Harrington] - Update Libpipeline to 1.3.1. Fixes ticket 994.
 - [William Harrington] - Update Make to 4.1. Fixes ticket #996.
 - [William Harrington] - Add Man-DB Home Page. Fixes ticket #997.
 - [William Harrington] - Update Man-DB to 2.7.0.2. Fixes ticket #998.
 - [William Harrington] - Update Man-pages to 3.75. Fixes ticket #999.
 - [William Harrington] - Update Perl to 5.20.1. Fixes ticket #1000. All hail the 1000 ticket!
 - [William Harrington] - Update Procs-ng to 3.3.10. Fixes ticket #1001.
 - [William Harrington] - Update Tar to 1.28. Fixes ticket #1004.
 - [William Harrington] - Update TCL to 8.6.2. Fixes ticket #1005.
- 18 October 2014

- [William Harrington] - Update LESS to 464.
- 18 October 2014
 - [William Harrington] - Changelog restarted, see the 3.0.0 book for the old changelog.

1.5. Changelog for MIPS 32 Bit

Below is a list of changes specific for this architecture made since the previous release of the book. For general changes see Master Changelog,

Changelog Entries:

- 18 October 2014
 - [William Harrington] - Changelog restarted, see the 3.0.0 book for the old changelog.

1.6. Resources

1.6.1. FAQ

If during the building of the CLFS 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://trac.clfs.org/wiki/faq>.

1.6.2. Mailing Lists

The `clfs.org` server hosts a number of mailing lists used for the development of the CLFS project. These lists include the main development and support lists, among others. If the FAQ does not contain your answer, you can search the CLFS lists via The Mail Archive <http://www.mail-archive.com>. You can find the mail lists with the following link:

<http://www.mail-archive.com/index.php?hunt=clfs>

For information on the different lists, how to subscribe, archive locations, and additional information, visit <http://trac.clfs.org/wiki/lists>.

1.6.3. News Server

Cross-LFS does not maintain its own News Server, but we do provide access via `gmane.org` <http://gmane.org>. If you want to subscribe to the Cross-LFS lists via a newsreader you can utilize `gmane.org`. You can find the `gmane` search for CLFS with the following link:

<http://dir.gmane.org/search.php?match=clfs>

1.6.4. IRC

Several members of the CLFS community offer assistance on our community Internet Relay Chat (IRC) network. Before using this support, please make sure that your question is not already answered in the CLFS FAQ or the mailing list archives. You can find the IRC network at `chat.freenode.net`. The support channel for cross-lfs is named `#cross-lfs`. If you need to show people the output of your problems, please use <http://pastebin.clfs.org> and reference the pastebin URL when asking your questions.

1.6.5. Mirror Sites

The CLFS project has a number of world-wide mirrors to make accessing the website and downloading the required packages more convenient. Please visit the CLFS website at <http://trac.clfs.org/wiki/mirrors> for mirrors of CLFS.

1.6.6. Contact Information

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

1.7. Help

If an issue or a question is encountered while working through this book, check the FAQ page at <http://trac.clfs.org/wiki/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://hints.clfs.org/index.php/Errors>.

We also have a wonderful CLFS community that is willing to offer assistance through the mailing lists and IRC (see the Section 1.6, “Resources” section of this book). However, we get several support questions everyday 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. This 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.

1.7.1. 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 GIT-20170803)
- The host distribution and version being used to create CLFS.
- The architecture of the host and target.
- The value of the `${CLFS_HOST}` and `${CLFS_TARGET}` environment variables, and if applicable, `${BUILD32}`, `${BUILD64}`, `${BUILDN32}`, and `${GCCTARGET}`.
- The package or section in which the problem was encountered.
- The exact error message or symptom received. See Section 1.7.3, “Compilation Problems” below for an example.
- Note whether you have deviated from the book at all. A package version change or even a minor change to any command is considered deviation.

Note

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

1.7.2. Configure Script Problems

If something goes wrong while running the **configure** script, review the `config.log` file. This file may contain the errors you encountered during **configure**. It often logs errors that may have not been printed to the screen. Include only the *relevant* lines if you need to ask for help.

1.7.3. 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/clfs/usr/share/locale:.\"
-DLOCALEDIR=\"/mnt/clfs/usr/share/locale\"
-DLIBDIR=\"/mnt/clfs/usr/lib\"
-DINCLUDEDIR=\"/mnt/clfs/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':
/clfs/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 `/clfs/tmp/make-3.79.1'
make[1]: *** [all-recursive] Error 1
make[1]: Leaving directory `/clfs/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 CLFS system is prepared. We will create the partition itself, create a file system on it, and mount it.

2.2. Setting The `${CLFS}` Variable

Throughout this book, the environment variable `CLFS` will be used several times. You should ensure that this variable is always defined throughout the CLFS build process. It should be set to the name of the directory where you will be building your CLFS system - we will use `/mnt/clfs` as an example, but the directory choice is up to you. If you are building CLFS on a separate partition, this directory will be the mount point for the partition. Choose a directory location and set the variable with the following command:

```
export CLFS=/mnt/clfs
```

Having this variable set is beneficial in that commands such as `install -dv ${CLFS}/tools` can be typed literally. The shell will automatically replace “`${CLFS}`” with “`/mnt/clfs`” (or whatever the variable was set to) when it processes the command line.

Do not forget to check that `${CLFS}` is set whenever you leave and reenter the current working environment (such as when doing a `su` to `root` or another user). Check that the `CLFS` variable is set up properly with:

```
echo ${CLFS}
```

Make sure the output shows the path to your CLFS system's build location, which is `/mnt/clfs` if the provided example was followed. If the output is incorrect, use the command given earlier on this page to set `${CLFS}` to the correct directory name.

2.3. Creating a New Partition

Like most other operating systems, CLFS is usually installed on a dedicated partition. The recommended approach to building a CLFS system is to use an available empty partition or, if you have enough unpartitioned space, to create one. However, if you're building for a different architecture you can simply build everything in “`/mnt/clfs`” (or whatever directory you want to use) and transfer it to your target machine. If you do not plan to use a separate partition for building CLFS, you can skip the rest of this chapter and continue on to Packages and Patches.

A minimal system requires around 6 gigabytes (GB). This is enough to store all the source tarballs and compile the packages. The CLFS 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. If the CLFS system is intended to be the primary Linux system, additional software will probably be installed which will require additional space (2-10 GB).

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 a CLFS system can be the same as the one used by the host system, in which case it is not necessary to create another one.

As the `root` user, start a disk partitioning program such as **cfdisk** or **fdisk** with a command line option naming the hard disk on which the new partition will be created—for example `/dev/sda` for the primary Integrated Drive Electronics (IDE) or Serial ATA (SATA) 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.

Remember the designation of the new partition (e.g., `sda5`). This book will refer to this as the CLFS partition. Also, remember the designation of the swap partition. These names will be needed later for the `/etc/fstab` file.

On a Cobalt RaQ2/Cube2 we use the existing firmware for a boot loader—it requires an `ext2` revision 0 partition to boot from. Here is the recommended partition layout for a Cobalt RaQ2/Cube2 system:

- The first partition should be 50-100 MB.
- The second partition should be all the remaining space minus the amount of RAM in the system.
- The third partition is going to be your swap partition, which will be the same size as the amount as the RAM installed in the system.

2.4. 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. We will create an `ext2` file system. Instructions for other file systems can be found at http://cblfs.clfs.org/index.php?section=6#File_System.

To create an `ext2` file system on the CLFS partition, run the following as `root`:

```
mke2fs /dev/[xxx]
```

Replace `[xxx]` with the name of the CLFS partition (`sda5` 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 CLFS system, as those features will not be supported by the CLFS-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: `dir_index`; `ext_attr`; `filetype`; `large_file`; `resize_inode` or `sparse_super` 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 CLFS partition. To do this, run the following commands as `root`:

```
cd /tmp
tar xjf /path/to/sources/e2fsprogs-1.43.4.tar.bz2
cd e2fsprogs-1.43.4
mkdir build
cd build
../configure
make #note that we intentionally don't 'make install' here!
./misc/mke2fs /dev/[xxx]
cd /tmp
rm -rf e2fsprogs-1.43.4
```

If you created a swap partition, you will need to initialize it for use by issuing the command below as `root`:

```
mkswap /dev/[yyy]
```

Replace `[yyy]` with the name of the swap partition. If you are using an existing swap partition, there is no need to format it.

The commands listed below are specific to the Cobalt MIPS systems, they have a special requirement to have a `ext2` Revision 0 for the boot partition. To make sure you satisfy this requirement, use the commands listed:

```
mke2fs -r 0 /dev/hda1
mke2fs /dev/hda2
mkswap /dev/hda3
```

2.5. 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.

As the `root` user, ensure the CLFS variable is set, if you haven't already:

```
export CLFS=/mnt/clfs
```

Next, create the mount point and mount the CLFS file system by running the following commands as `root`:

```
mkdir -pv ${CLFS}
mount -v /dev/[xxx] ${CLFS}
```

Replace `[xxx]` with the designation of the CLFS partition.

If using multiple partitions for CLFS (e.g., one for `/` and another for `/usr`), mount them as `root` using:

```
mkdir -pv ${CLFS}
mount -v /dev/[xxx] ${CLFS}
mkdir -v ${CLFS}/usr
mount -v /dev/[yyy] ${CLFS}/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`, `nodev`, or `noatime` options). Run `mount | grep ${CLFS}` to see what options are set for the mounted CLFS partition. If `nosuid`, `nodev`, and/or `noatime` are set, the partition will need to be remounted.

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 for building 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 not 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://clfs.org/files/packages/git/>.

Create a directory called `${CLFS}/sources` and use it to store your sources and patches. All packages should be compiled there as well. Using any other location for compiling may have unexpected results.

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

```
mkdir -v ${CLFS}/sources
```

Make this directory writable and sticky. When a directory is marked “sticky”, that means that even if multiple users have write permission on that directory, any file within that directory can only be deleted or modified by its owner. The following command, run as `root`, will enable the write and sticky modes:

```
chmod -v a+wt ${CLFS}/sources
```

You can download all needed packages and patches into this directory either by using the links on the following pages in this section, or by passing the *download list* to `wget`:

```
wget -i dl.list -P ${CLFS}/sources
```

Verification of downloaded packages can be done by downloading the following MD5 or SHA1 checksum lists:

MD5SUMS:

```
pushd ${CLFS}/sources
md5sum -c MD5SUMS
popd
```

SHA1SUMS:

```
pushd ${CLFS}/sources
sha1sum -c SHA1SUMS
popd
```

3.2. All Packages

Download or otherwise obtain the following packages:

• **Acl (2.2.52) - 384 KB:**

Home page: <http://savannah.nongnu.org/projects/acl>

Download: <http://download.savannah.gnu.org/releases/acl/acl-2.2.52.src.tar.gz>

MD5 sum: a61415312426e9c2212bd7dc7929abda

• **Attr (2.4.47) - 340 KB:**

Home page: <http://savannah.nongnu.org/projects/attr>

Download: <http://download.savannah.gnu.org/releases/attr/attr-2.4.47.src.tar.gz>

MD5 sum: 84f58dec00b60f2dc8fd1c9709291cc7

• **Autoconf (2.69) - 1,188 KB:**

Home page: <http://www.gnu.org/software/autoconf/autoconf.html>

Download: <http://ftp.gnu.org/gnu/autoconf/autoconf-2.69.tar.xz>

MD5 sum: 50f97f4159805e374639a73e2636f22e

• **Automake (1.15) - 1,497 KB:**

Home page: <http://www.gnu.org/software/automake>

Download: <http://ftp.gnu.org/gnu/automake/automake-1.15.tar.xz>

MD5 sum: 9a1ddb0e053474d9d1105cfe39b0c48d

• **Bash (4.4) - 9,377 KB:**

Home page: <http://www.gnu.org/software/bash>

Download: <http://ftp.gnu.org/gnu/bash/bash-4.4.tar.gz>

MD5 sum: 148888a7c95ac23705559b6f477dfe25

• **Bc (1.07.1) - 420 KB:**

Home page: <http://www.gnu.org/software/bc/>

Download: <http://ftp.gnu.org/gnu/bc/bc-1.07.1.tar.gz>

MD5 sum: cda93857418655ea43590736fc3ca9fc

• **Binutils (2.28) - 26,556 KB:**

Home page: <http://sources.redhat.com/binutils>

Download: <http://ftp.gnu.org/gnu/binutils/binutils-2.28.tar.bz2>

MD5 sum: 9e8340c96626b469a603c15c9d843727

• **Bison (3.0.4) - 1,974 KB:**

Home page: <http://www.gnu.org/software/bison>

Download: <http://ftp.gnu.org/gnu/bison/bison-3.0.4.tar.xz>

MD5 sum: c342201de104cc9ce0a21e0ad10d4021

• **Boot-scripts for CLFS (3.0-20140710) - 22 KB:**

Download: <http://clfs.org/files/packages/git/boot-scripts-cross-lfs-3.0-20140710.tar.xz>

MD5 sum: bac3e8a54e5ab124a2df0713dc4e4ca4

• **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 (0.11.0) - 754 KB:**

Home page: <http://libcheck.github.io/check/>

Download: <https://github.com/libcheck/check/releases/download/0.11.0/check-0.11.0.tar.gz>

MD5 sum: 9b90522b31f5628c2e0f55dda348e558

• **CLFS Network Scripts (20140224) - 22 KB:**

Download: <http://clfs.org/files/clfs-network-scripts-20140224.tar.xz>

MD5 sum: 831308d5e80bdaa3f494dc218ee43f78

• Coreutils (8.27) - 5,286 KB:Home page: <http://www.gnu.org/software/coreutils/coreutils.html>Download: <http://ftp.gnu.org/gnu/coreutils/coreutils-8.27.tar.xz>

MD5 sum: 502795792c212932365e077946d353ae

• D-Bus (1.10.18) - 1,987 KB:Home page: <http://www.freedesktop.org/wiki/Software/dbus>Download: <http://dbus.freedesktop.org/releases/dbus/dbus-1.10.18.tar.gz>

MD5 sum: 1209c455598165a0c5263d4201894179

• DejaGNU (1.6) - 524 KB:Home page: <http://www.gnu.org/software/dejagnu>Download: <http://ftp.gnu.org/gnu/dejagnu/dejagnu-1.6.tar.gz>

MD5 sum: 1fdc2eb0d592c4f89d82d24dfdf02f0b

• DHCPD (6.11.5) - 198 KB:Home page: <http://roy.marples.name/projects/dhcpd>Download: <http://roy.marples.name/downloads/dhcpd/dhcpd-6.11.5.tar.xz>

MD5 sum: 2465624b62c1154f0e89dc69c42c849b

• Diffutils (3.6) - 1,398 KB:Home page: <http://www.gnu.org/software/diffutils>Download: <http://ftp.gnu.org/gnu/diffutils/diffutils-3.6.tar.xz>

MD5 sum: 07cf286672ced26fba54cd0313bdc071

• E2fsprogs (1.43.4) - 5,266 KB:Home page: <http://e2fsprogs.sourceforge.net>Download: <http://www.kernel.org/pub/linux/kernel/people/tytso/e2fsprogs/v1.43.4/e2fsprogs-1.43.4.tar.xz>

MD5 sum: 8903379ef0b902f4e29b6cafea359fe1

• Eudev (1.7) - 1,756 KB:Home page: <https://wiki.gentoo.org/wiki/Eudev>Download: <http://dev.gentoo.org/~blueness/eudev/eudev-1.7.tar.gz>

MD5 sum: 80649a0350ff9620fc2da9562d9f2a6a

• Expat (2.2.0) - 414 KB:Home page: <http://expat.sourceforge.net>Download: <http://downloads.sourceforge.net/expat/expat-2.2.0.tar.bz2>

MD5 sum: 2f47841c829facb346eb6e3fab5212e2

• Expect (5.45) - 616 KB:Home page: <http://expect.sourceforge.net>Download: <http://downloads.sourceforge.net/project/expect/Expect/5.45/expect5.45.tar.gz>

MD5 sum: 44e1a4f4c877e9ddc5a542dfa7ecc92b

• File (5.31) - 792 KB:Home page: <http://www.darwinsys.com/file>Download: <ftp://ftp.astron.com/pub/file/file-5.31.tar.gz>

MD5 sum: 319627d20c9658eae85b056115b8c90a

Note

File (5.31) 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 is <http://clfs.org/files/packages/git/>.

• **Findutils (4.6.0) - 3,780 KB:**

Home page: <http://www.gnu.org/software/findutils>

Download: <http://ftp.gnu.org/gnu/findutils/findutils-4.6.0.tar.gz>

MD5 sum: 9936aa8009438ce185bea2694a997fc1

• **Flex (2.6.4) - 1,419 KB:**

Home page: <https://github.com/westes/flex>

Download: <https://github.com/westes/flex/releases/download/v2.6.4/flex-2.6.4.tar.gz>

MD5 sum: 2882e3179748cc9f9c23ec593d6adc8d

• **Gawk (4.1.4) - 2,368 KB:**

Home page: <http://www.gnu.org/software/gawk>

Download: <http://ftp.gnu.org/gnu/gawk/gawk-4.1.4.tar.xz>

MD5 sum: 4e7dbc81163e60fd4f0b52496e7542c9

• **GCC (7.1.0) - 84,304 KB:**

Home page: <http://gcc.gnu.org>

Download: <ftp://gcc.gnu.org/pub/gcc/releases/gcc-7.1.0/gcc-7.1.0.tar.bz2>

MD5 sum: 6bf56a2bca9dac9dbbf8e8d1036964a8

• **GDBM (1.13) - 892 KB:**

Home page: <http://www.gnu.org/software/gdbm>

Download: <http://ftp.gnu.org/gnu/gdbm/gdbm-1.13.tar.gz>

MD5 sum: 8929dcda2a8de3fd2367bdbf66769376

• **Gettext (0.19.8.1) - 7,210 KB:**

Home page: <http://www.gnu.org/software/gettext>

Download: <http://ftp.gnu.org/gnu/gettext/gettext-0.19.8.1.tar.xz>

MD5 sum: df3f5690eaa30fd228537b00cb7b7590

• **Glibc (2.25) - 13,874 KB:**

Home page: <http://www.gnu.org/software/libc/>

Download: <http://ftp.gnu.org/gnu/glibc/glibc-2.25.tar.xz>

MD5 sum: 1496c3bf41adf9db0ebd0af01f202eed

• **GMP (6.1.2) - 1,946 KB:**

Home page: <https://gmplib.org/>

Download: <http://ftp.gnu.org/gnu/gmp/gmp-6.1.2.tar.xz>

MD5 sum: f58fa8001d60c4c77595fbbb62b63c1d

• **Gperf (3.0.4) - 968 KB:**

Home page: <http://www.gnu.org/software/gperf>

Download: <http://ftp.gnu.org/gnu/gperf/gperf-3.0.4.tar.gz>

MD5 sum: c1f1db32fb6598d6a93e6e88796a8632

• **Grep (3.0) - 1,375 KB:**

Home page: <http://www.gnu.org/software/grep>

Download: <http://ftp.gnu.org/gnu/grep/grep-3.0.tar.xz>

MD5 sum: fa07c1616adeb9c3262be5177d10ad4a

• **Groff (1.22.3) - 4,189 KB:**

Home page: <http://www.gnu.org/software/groff>

Download: <http://ftp.gnu.org/gnu/groff/groff-1.22.3.tar.gz>

MD5 sum: cc825fa64bc7306a885f2fb2268d3ec5

• **Gzip (1.8) - 728 KB:**

Home page: <http://www.gnu.org/software/gzip/gzip.html>

Download: <http://ftp.gnu.org/gnu/gzip/gzip-1.8.tar.xz>

MD5 sum: f7caabb65cddc1a4165b398009bd05b9

• **Iana-Etc (2.30) - 204 KB:**

Home page: <https://www.archlinux.org/packages/core/any/iana-etc/>

Download: <http://ftp.clfs.org/pub/clfs/conglomeration/iana-etc/iana-etc-2.30.tar.bz2>

MD5 sum: 3ba3afb1d1b261383d247f46cb135ee8

• **Intltool (0.51.0) - 162 KB:**

Home page: <http://freedesktop.org/wiki/Software/intltool>

Download: <http://launchpad.net/intltool/trunk/0.51.0/+download/intltool-0.51.0.tar.gz>

MD5 sum: 12e517cac2b57a0121cda351570f1e63

• **IPRoute2 (4.9.0) - 613 KB:**

Home page: <http://www.linuxfoundation.org/collaborate/workgroups/networking/iproute2>

Download: <http://www.kernel.org/pub/linux/utils/net/iproute2/iproute2-4.9.0.tar.xz>

MD5 sum: 44a8371a4b2c40e48e4c9f98cbd41391

• **IPutils (s20150815) - 152 KB:**

Home page: <http://www.linuxfoundation.org/collaborate/workgroups/networking/iputils>

Download: <http://clfs.org/files/packages/git/iputils-s20150815.tar.xz>

MD5 sum: d184faea97095265452dce19ef98daf6

• **ISL (0.17.1) - 1,441 KB:**

Home page: <http://isl.gforge.inria.fr>

Download: <http://isl.gforge.inria.fr/isl-0.17.1.tar.xz>

MD5 sum: 20b83900e234f982a566a3a6b3503bf1

• **Kbd (2.0.4) - 1,019 KB:**

Home page: <http://ftp.altlinux.org/pub/people/legion/kbd/>

Download: <http://www.kernel.org/pub/linux/utils/kbd/kbd-2.0.4.tar.xz>

MD5 sum: c1635a5a83b63aca7f97a3eab39ebaa6

• **Kmod (24) - 537 KB:**

Home page: <http://git.kernel.org/?p=utils/kernel/kmod/kmod.git;a=summary>

Download: <http://www.kernel.org/pub/linux/utils/kernel/kmod/kmod-24.tar.xz>

MD5 sum: 08297dfb6f2b3f625f928ca3278528af

• **Less (491) - 320 KB:**

Home page: <http://www.greenwoodsoftware.com/less>

Download: <http://www.greenwoodsoftware.com/less/less-491.tar.gz>

MD5 sum: 81e260e8b12f253c31565acad6ee0e59

• **Libcap (2.25) - 64 KB:**

Home page: <http://sites.google.com/site/fullycapable/>

Download: <http://www.kernel.org/pub/linux/libs/security/linux-privs/libcap2/libcap-2.25.tar.xz>

MD5 sum: 6666b839e5d46c2ad33fc8aa2ceb5f77

• **Libpipeline (1.4.1) - 805 KB:**

Home page: <http://libpipeline.nongnu.org/>

Download: <http://download.savannah.gnu.org/releases/libpipeline/libpipeline-1.4.1.tar.gz>

MD5 sum: e54590ec68d6c1239f67b5b44e92022c

• **Libtool (2.4.6) - 948 KB:**

Home page: <http://www.gnu.org/software/libtool>

Download: <http://ftp.gnu.org/gnu/libtool/libtool-2.4.6.tar.xz>

MD5 sum: 1bfb9b923f2c1339b4d2ce1807064aa5

• **Linux (4.9) - 93,192 KB:**

Home page: <http://www.kernel.org>

Download: <http://www.kernel.org/pub/linux/kernel/v4.x/linux-4.9.tar.xz>

MD5 sum: 0a68ef3615c64bd5ee54a3320e46667d

• **M4 (1.4.18) - 1,208 KB:**

Home page: <http://www.gnu.org/software/m4/m4.html>

Download: <http://ftp.gnu.org/gnu/m4/m4-1.4.18.tar.xz>

MD5 sum: 730bb15d96fffe47e148d1e09235af82

• **Make (4.2.1) - 1,407 KB:**

Home page: <http://www.gnu.org/software/make>

Download: <http://ftp.gnu.org/gnu/make/make-4.2.1.tar.bz2>

MD5 sum: 15b012617e7c44c0ed482721629577ac

• **Man-DB (2.7.6.1) - 1,541 KB:**

Home page: <http://man-db.nongnu.org>

Download: <http://cifs.org/files/packages/git/man-db-2.7.6.1.tar.xz>

MD5 sum: 2948d49d0ed7265f60f83aa4a9ac9268

• **Man-pages (4.09) - 1,522 KB:**

Home page: <http://www.win.tue.nl/~aeb/linux/man>

Download: <http://www.kernel.org/pub/linux/docs/man-pages/man-pages-4.09.tar.xz>

MD5 sum: 91c721409bbf823d8f62bee3a1fe8ae3

• **MPC (1.0.3) - 670 KB:**

Home page: <http://www.multiprecision.org/>

Download: <http://www.multiprecision.org/mpc/download/mpc-1.0.3.tar.gz>

MD5 sum: d6a1d5f8ddea3abd2cc3e98f58352d26

• **MPFR (3.1.5) - 1,127 KB:**

Home page: <http://www.mpfr.org/>

Download: <http://www.mpfr.org/mpfr-3.1.5/mpfr-3.1.5.tar.xz>

MD5 sum: c4ac246cf9795a4491e7766002cd528f

- **Ncurses (6.0) - 3,132 KB:**

Home page: <http://www.gnu.org/software/ncurses>

Download: <http://ftp.gnu.org/gnu/ncurses/ncurses-6.0.tar.gz>

MD5 sum: ee13d052e1ead260d7c28071f46eefb1

- **Patch (2.7.5) - 728 KB:**

Home page: <http://savannah.gnu.org/projects/patch>

Download: <http://ftp.gnu.org/gnu/patch/patch-2.7.5.tar.xz>

MD5 sum: e3da7940431633fb65a01b91d3b7a27a

- **Perl (5.26.0) - 11,962 KB:**

Home page: <https://www.perl.org>

Download: <http://www.cpan.org/src/5.0/perl-5.26.0.tar.xz>

MD5 sum: 8c6995718e4cb62188f0d5e3488cd91f

- **Pkg-config-lite (0.28-1) - 384 KB:**

Home page: <http://sourceforge.net/projects/pkgconfiglite>

Download: <http://sourceforge.net/projects/pkgconfiglite/files/0.28-1/pkg-config-lite-0.28-1.tar.gz>

MD5 sum: 61f05feb6bab0a6bbfab4b6e3b2f44b6

- **Procps-ng (3.3.12) - 845 KB:**

Home page: <http://sourceforge.net/projects/procps-ng>

Download: <http://sourceforge.net/projects/procps-ng/files/Production/procps-ng-3.3.12.tar.xz>

MD5 sum: 957e42e8b193490b2111252e4a2b443c

- **Psmisc (22.21) - 458 KB:**

Home page: <http://psmisc.sourceforge.net>

Download: <http://downloads.sourceforge.net/psmisc/psmisc-22.21.tar.gz>

MD5 sum: 935c0fd6eb208288262b385fa656f1bf

- **Readline (7.0) - 2,910 KB:**

Home page: <http://cnswww.cns.cwru.edu/php/chet/readline/rltop.html>

Download: <http://ftp.gnu.org/gnu/readline/readline-7.0.tar.gz>

MD5 sum: 205b03a87fc83dab653b628c59b9fc91

- **Sed (4.4) - 1,182 KB:**

Home page: <http://www.gnu.org/software/sed>

Download: <http://ftp.gnu.org/gnu/sed/sed-4.4.tar.xz>

MD5 sum: e0c583d4c380059abd818cd540fe6938

- **Shadow (4.5) - 1,627 KB:**

Home page: <http://pkg-shadow.alioth.debian.org>

Download: <https://github.com/shadow-maint/shadow/releases/download/4.5/shadow-4.5.tar.xz>

MD5 sum: c350da50c2120de6bb29177699d89fe3

- **Sysvinit (2.88dsf) - 104 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

- **Systemd (233) - 4,106 KB:**

Home page: <http://freedesktop.org/wiki/Software/systemd>

Download: <http://www.linuxfromscratch.org/~krejzi/systemd/systemd-233.tar.xz>

MD5 sum: e67d7664714ff5efe5b59cddedbe26a4

- **Tar (1.29) - 1,996 KB:**

Home page: <http://www.gnu.org/software/tar>

Download: <http://ftp.gnu.org/gnu/tar/tar-1.29.tar.xz>

MD5 sum: a1802fec550baaeecff6c381629653ef

- **Tcl (8.6.4) - 8,857 KB:**

Home page: <http://www.tcl.tk>

Download: <http://downloads.sourceforge.net/tcl/tcl-core8.6.4-src.tar.gz>

MD5 sum: 8b8c9d85469d8dbe32e51117b8ef11e3

- **Texinfo (6.3) - 4,468 KB:**

Home page: <http://www.gnu.org/software/texinfo>

Download: <http://ftp.gnu.org/gnu/texinfo/texinfo-6.3.tar.xz>

MD5 sum: 32baefe5c7080dfb512a4eac5ce67b2a

- **Time Zone Data (2017b) - 324 KB:**

Home page: <http://www.iana.org/time-zones>

Download: <http://www.iana.org/time-zones/repository/releases/tzdata2017b.tar.gz>

MD5 sum: 50dc0dc50c68644c1f70804f2e7a1625

- **Util-linux (2.29.2) - 4,278 KB:**

Download: <http://www.kernel.org/pub/linux/utils/util-linux/v2.29/util-linux-2.29.2.tar.xz>

MD5 sum: 63c40c2068fcbb7e1d5c1d281115d973

- **Vim (8.0) - 10,867 KB:**

Home page: <http://www.vim.org>

Download: <ftp://ftp.vim.org/pub/vim/unix/vim-8.0.tar.bz2>

MD5 sum: b35e794140c196ff59b492b56c1e73db

- **XML::Parser (2.44) - 237 KB:**

Home page: <https://github.com/chorny/XML-Parser>

Download: <http://search.cpan.org/CPAN/authors/id/T/TO/TODDR/XML-Parser-2.44.tar.gz>

MD5 sum: af4813fe3952362451201ced6fbce379

- **XZ Utils (5.2.3) - 1,032 KB:**

Home page: <http://tukaani.org/xz/>

Download: <http://tukaani.org/xz/xz-5.2.3.tar.xz>

MD5 sum: 60fb79cab777e3f71ca43d298adacbd5

- **Zlib (1.2.11) - 468 KB:**

Home page: <http://www.zlib.net>

Download: <http://zlib.net/zlib-1.2.11.tar.xz>

MD5 sum: 85adef240c5f370b308da8c938951a68

Note

Zlib (1.2.11) 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 is <http://clfs.org/files/packages/git/>.

Total size of these packages: about 343 MB

3.3. Additional Packages for MIPS

- **ARCLoad (0.5) - 48 KB:**

Home page: <http://www.linux-mips.org/wiki/ARCLoad>

Download: <ftp://ftp.linux-mips.org/pub/linux/mips/people/skylark/arclload-0.5.tar.bz2>

MD5 sum: b00e1c79074a13c2de97748f56f9bd1f

- **Colo (1.22) - 252 KB:**

Home page: <http://www.colonel-panic.org/cobalt-mips>

Download: <http://www.colonel-panic.org/cobalt-mips/colo/colo-1.22.tar.gz>

MD5 sum: 52c16ad31f3b88f710f0fdb5abed0457

- **DVHtool (1.0.1) - 56 KB:**

Home page: <http://packages.qa.debian.org/d/dvhtool.html>

Download: http://ftp.debian.org/debian/pool/main/d/dvhtool/dvhtool_1.0.1.orig.tar.gz

MD5 sum: 4448c01e6a015685af90a79fba8da4e

Total size of these packages: about 356 KB

3.4. 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 a CLFS system:

- **Automake Perl Patch - 0.685 KB:**

Download: http://patches.clfs.org/dev/automake-1.15-perl_5_26-1.patch

MD5 sum: 99dd9d0a31f4a51fbe77bbc13aa9d783

- **Bash Branch Update Patch - 17 KB:**

Download: http://patches.clfs.org/dev/bash-4.4-branch_update-1.patch

MD5 sum: 9f59bec94bfd1023d9c6a76b49187420

- **Coreutils Uname Patch - 5.020 KB:**

Download: <http://patches.clfs.org/dev/coreutils-8.27-uname-1.patch>

MD5 sum: e9bc0459d313f677cebc2095c3b18818

- **Iana-Etc Protocol and Port Numbers Update - 282 KB:**

Download: http://patches.clfs.org/dev/iana-etc-2.30-numbers_update-20140202-2.patch.xz

MD5 sum: b0e7051fef0b3ba064209a5f3d23bd2a

- **Intool Perl Fix - 2.390 KB:**

Download: <http://patches.clfs.org/dev/intltool-0.51.0-perl-5.22-compatibility.patch>

MD5 sum: 476b85e0c100eb2c8fbd74825cb4761e

- **IPUtils Fixes Patch - 47 KB:**

Download: <http://patches.clfs.org/dev/iputils-s20150815-build-1.patch>

MD5 sum: 113c166a13d33a3ec10e1ec65542bdc3

- **Linux Sublevel Patch - 465 KB:**

Download: <http://www.kernel.org/pub/linux/kernel/v4.x/patch-4.9.21.xz>

MD5 sum: 59351116e4dfdb9072dd8cccd15e1800

- **MPFR Fixes Patch - 5.101 KB:**

Download: <http://patches.clfs.org/dev/mpfr-3.1.5-fixes-1.patch>

MD5 sum: c9e1bfc93ee8d90226772b628ab77f38

- **Ncurses Bash Patch - .743 KB:**

Download: http://patches.clfs.org/dev/ncurses-6.0-bash_fix-1.patch

MD5 sum: e3b6d45ce0f0b87e0df98e5bc0d09415

- **Readline Branch Update - 3.049 KB:**

Download: http://patches.clfs.org/dev/readline-7.0-branch_update-1.patch

MD5 sum: 8b429202ce52362ae927f36a48461c23

- **Sysvinit Tools Updates Patch - 2.339 KB:**

Download: http://patches.clfs.org/dev/sysvinit-2.88dsf-tools_updates-1.patch

MD5 sum: c3f6981c46868b68bfd58921570ea51f

- **Vim Branch Update Patch - 961 KB:**

Download: http://patches.clfs.org/dev/vim-8.0-branch_update-1.patch

MD5 sum: 5a076d526aee80e1d32decf46b5287f9

Total size of these patches: about 1,791.327 KB

In addition to the above required patches, there exist a number of optional patches created by the CLFS 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://patches.clfs.org/dev/> and acquire any additional patches to suit the system needs.

3.5. Additional Patches for MIPS

- **Colo Relocation Patch - .749 KB:**

Download: http://patches.clfs.org/dev/colo-1.22-relocation_fix-1.patch

MD5 sum: e0607ee1071f2f805ffa1ef1c5b1a766

- **Dvhtool Fixes - 6.3 KB:**

Download: <http://patches.clfs.org/dev/dvhtool-1.0.1-fixes-1.patch>

MD5 sum: a521b380354b6a0c96b2d6308372749d

- **GCC Specs Patch - 21 KB:**

Download: <http://patches.clfs.org/dev/gcc-7.1.0-specs-1.patch>

MD5 sum: 9f4efe1ae2cb46e6ad41b2ba2a24e5d8

Total size of these patches: about 28.049 KB

Chapter 4. Final Preparations

4.1. Introduction

In this chapter, we will perform a few additional tasks to prepare for building the cross-compile tools. We will create directories in `${CLFS}` for the installation of the cross-toolchain and temporary system, add an unprivileged user to reduce risk, and create an appropriate build environment for that user.

4.2. Creating the `${CLFS}/tools` Directory

All programs compiled in Constructing a Temporary System will be installed under `${CLFS}/tools` to keep them separate from the programs compiled in Installing Basic System Software. The programs compiled here are temporary tools and will not be a part of the final CLFS 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 Constructing a Temporary System).

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

```
install -dv ${CLFS}/tools
```

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

```
ln -sv ${CLFS}/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. This will provide a common place for our temporary tools system.

4.3. Creating the `${CLFS}/cross-tools` Directory

The cross-binutils and cross-compiler built in Constructing Cross-Compile Tools will be installed under `${CLFS}/cross-tools` to keep them separate from the host programs. The programs compiled here are cross-tools and will not be a part of the final CLFS system or the temp-system. By keeping these programs in a separate directory, they can easily be discarded later after their use.

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

```
install -dv ${CLFS}/cross-tools
```

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

```
ln -sv ${CLFS}/cross-tools /
```

4.4. Adding the CLFS User

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

```
groupadd clfs
useradd -s /bin/bash -g clfs -d /home/clfs clfs
mkdir -pv /home/clfs
chown -v clfs:clfs /home/clfs
```

The meaning of the command line options:

`-s /bin/bash`

This makes **bash** the default shell for user `clfs`.

Important

The build instructions assume that the **bash** shell is in use.

`-g clfs`

This option adds the new user to the `clfs` group.

`-d /home/clfs`

This option sets the user's home directory, but does not create it. We could have used `-m` to tell **useradd** to create the directory as well, but this would also copy the contents of the host system's `/etc/skel` directory into the new user's home. We would prefer to have a clean user environment, so we just create an empty directory after adding the user.

`clfs`

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

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

```
passwd clfs
```

As `root`, grant `clfs` full access to `${CLFS}/cross-tools` and `${CLFS}/tools` by making `clfs` the directories' owner:

```
chown -v clfs ${CLFS}/tools
chown -v clfs ${CLFS}/cross-tools
```

If a separate working directory was created as suggested, run the following command as `root` to give user `clfs` ownership of this directory as well:

```
chown -v clfs ${CLFS}/sources
```

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

```
su - clfs
```

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**.

Note

Until specified otherwise, all commands from this point on should be done as the `clfs` user.

4.5. 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 `clfs`, 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 `clfs`, 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
CLFS=/mnt/clfs
LC_ALL=POSIX
PATH=/cross-tools/bin:/bin:/usr/bin
export CLFS LC_ALL PATH
unset CFLAGS CXXFLAGS PKG_CONFIG_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 `/cross-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 `CLFS` 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. Setting `LC_ALL` to “POSIX” or “C” (the two are equivalent) ensures that everything will work as expected in the temporary build environment.

By putting `/cross-tools/bin` at the beginning of the `PATH`, the cross-compiler built in Constructing Cross-Compile Tools will be picked up by the build process for the temp-system packages before anything that may be installed on the host. This, combined with turning off hashing, helps to ensure that you will be using the cross-compile tools to build the temp-system in `/tools`.

The `CFLAGS`, `CXXFLAGS` and `PKG_CONFIG_PATH` variables should not be set while building the temporary system, so we unset them.

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

```
source ~/.bash_profile
```

4.6. Build Variables

Setting Host and Target

During the building of the cross-compile tools you will need to set a few variables that will be dependent on your particular needs. The first variable will be the triplet of the host machine, which will be put into the `CLFS_HOST` variable. To account for the possibility that the host and target are the same arch, as cross-compiling won't work when host and target are the same, part of the triplet needs to be changed slightly - in our case, we will change part of the triplet to "cross". Set `CLFS_HOST` using the following command:

```
export CLFS_HOST=$(echo ${MACHTYPE} | sed -e 's/-[^-]*-/cross/')
```

Now you will need to set the triplet for the target architecture. Set the target variable using the following command:

For a MIPS Little Endian Machine:

```
export CLFS_TARGET="mipsel-unknown-linux-gnu"
```

For a MIPS Big Endian Machine:

```
export CLFS_TARGET="mips-unknown-linux-gnu"
```

Copy settings to Environment

Now add these to `~/ .bashrc`, just in case you have to exit and restart building later:

```
cat >> ~/.bashrc << EOF
export CLFS_HOST="${CLFS_HOST}"
export CLFS_TARGET="${CLFS_TARGET}"
EOF
```

4.7. About the Test Suites

Most packages provide a test suite, usually a script or **make** target, which tests the just-compiled programs or libraries by executing or linking to them. Test suites are often useful for verifying that a package compiled correctly. However, they cannot be run while cross-compiling so we will not mention test suite commands for any packages until Installing Basic System Software.

Part III. Make the Cross-Compile Tools

Chapter 5. Constructing Cross-Compile Tools

5.1. Introduction

This chapter shows you how to create cross platform tools.

If for some reason you have to stop and come back later, remember to use the **su - cifs** command, and it will setup the build environment that you left.

5.1.1. Common Notes

Important

Before issuing the build instructions for a package, the package should be unpacked, and a **cd** into the created directory should be performed.

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 chapters, 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.

Important

After installing each package, both in this and the next chapters, delete its source and build directories, unless specifically instructed otherwise. Deleting the sources prevents mis-configuration when the same package is reinstalled later.

5.2. File-5.31

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

5.2.1. Installation of File

One method that **file** uses for identifying a given file is to run “magic tests”, where it compares the file's contents to data in “magic files”, which contain information about a number of standard file formats. When File is compiled, it will run **file -C** to combine the information from the magic files in its source tree into a single `magic.mgc` file, which it will use after it is installed. When we build File in Constructing a Temporary System, it will be cross-compiled, so it will not be able to run the **file** program that it just built, which means that we need one that will run on the host system.

Prepare File for compilation:

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

The meaning of the configure options:

--prefix=/cross-tools

This tells the configure script to prepare to install the package in the `/cross-tools` directory.

Compile the package:

```
make
```

Install the package:

```
make install
```

Details on this package are located in Section 10.42.2, “Contents of File.”

5.3. Linux-4.9.21 Headers

The Linux Kernel contains a **make** target that installs “sanitized” kernel headers.

5.3.1. Installation of Linux Headers

Note

For this step you will need to unpack the kernel tarball (`linux-4.9.tar.xz`) and **cd** into its source directory before entering the commands on this page.

Apply the latest Linux sublevel patch:

```
xzcat ../patch-4.9.21.xz | patch -Np1 -i -
```

Install the kernel header files:

```
make mrproper
make ARCH=mips headers_check
make ARCH=mips INSTALL_HDR_PATH=/tools headers_install
```

The meaning of the make commands:

make mrproper

Ensures that the kernel source dir is clean.

make ARCH=mips headers_check

Sanitizes the raw kernel headers so that they can be used by userspace programs.

make ARCH=mips INSTALL_HDR_PATH=/tools headers_install

This will install the kernel headers into `/tools/include`.

Details on this package are located in Section 10.5.2, “Contents of Linux Headers.”

5.4. M4-1.4.18

The M4 package contains a macro processor.

5.4.1. Installation of M4

M4 is required to build GMP. We will compile and install an **m4** program into `/cross-tools`, so that we have a known-good version which can be used to build GMP, both in Cross-Tools and the Temporary System.

Prepare M4 for compilation:

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

Compile the package:

```
make
```

Install the package:

```
make install
```

Details on this package are located in Section 10.9.2, “Contents of M4.”

5.5. Ncurses-6.0

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

5.5.1. Installation of Ncurses

When Ncurses is compiled, it executes **tic** to create a terminfo database in `${prefix}/share/terminfo`. If possible, the `Makefile` will use the **tic** binary that was just compiled in its source tree, but this does not work when Ncurses is cross-compiled. To allow the Ncurses build in Constructing a Temporary System to succeed, we will build and install a **tic** program that can be run on the host system.

Prepare Ncurses for compilation:

```
AWK=gawk ./configure \
  --prefix=/cross-tools \
  --without-debug
```

The meaning of the new configure options:

--without-debug

Tells Ncurses to build without debugging information.

Only one binary is needed for the Cross-Tools. Build the headers and then build **tic**:

```
make -C include
make -C progs tic
```

Install **tic** with the following command:

```
install -v -m755 progs/tic /cross-tools/bin
```

Details on this package are located in Section 10.24.2, “Contents of Ncurses.”

5.6. Pkg-config-lite-0.28-1

Pkg-config-lite is a tool to help you insert the correct compiler options on the command line when compiling applications and libraries.

5.6.1. Installation of Pkg-config-lite

Several packages in the temporary system will use **pkg-config** to find various required and optional dependencies. Unfortunately, this could result in those packages finding libraries on the host system and trying to link against them, which will not work. To avoid this problem, we will install a **pkg-config** binary in `/cross-tools` and configure it so that it will look for Pkg-config files only in `/tools`.

Prepare Pkg-config-lite for compilation:

```
./configure \
  --prefix=/cross-tools \
  --host=${CLFS_TARGET} \
  --with-pc-path=/tools/lib/pkgconfig:/tools/share/pkgconfig
```

The meaning of the new configure option:

`--host=${CLFS_TARGET}`

Several packages that we will cross-compile later will try to search for `${CLFS_TARGET}-pkg-config`. Setting this option ensures that Pkg-config-lite will create a hard link in `/cross-tools/bin` with this name, so that it will be used instead of any similarly-named program that might exist on the host.

`--with-pc-path`

This sets the default `PKG_CONFIG_PATH` to `/tools/lib/pkgconfig` and `/tools/share/pkgconfig`.

Compile the package:

```
make
```

Install the package:

```
make install
```

Details on this package are located in Section 10.23.2, “Contents of Pkg-config-lite.”

5.7. GMP-6.1.2

GMP is a library for arithmetic on arbitrary precision integers, rational numbers, and floating-point numbers.

5.7.1. Installation of GMP

This package and the next two - MPFR and MPC - will be installed into `/cross-tools` because GCC requires them to build.

Prepare GMP for compilation:

```
./configure \  
  --prefix=/cross-tools \  
  --enable-cxx \  
  --disable-static
```

The meaning of the new configure option:

`--enable-cxx`

This tells GMP to enable C++ support.

`--disable-static`

This tells the GMP package not to compile or install static libraries, which are not needed for the Cross-Tools

Compile the package:

```
make
```

Install the package:

```
make install
```

Details on this package are located in Section 10.10.2, “Contents of GMP.”

5.8. MPFR-3.1.5

The MPFR library is a C library for multiple-precision floating-point computations with correct rounding.

5.8.1. Installation of MPFR

Apply a patch with upstream fixes:

```
patch -Np1 -i ../mpfr-3.1.5-fixes-1.patch
```

Prepare MPFR for compilation:

```
LDFLAGS="-Wl,-rpath,/cross-tools/lib" \  
./configure \  
  --prefix=/cross-tools \  
  --disable-static \  
  --with-gmp=/cross-tools
```

The meaning of the new configure options:

```
LDFLAGS="-Wl,-rpath,/cross-tools/lib"
```

This tells **configure** to search in `/cross-tools` for libraries.

```
--with-gmp=/cross-tools
```

This tells **configure** where to find GMP.

Compile the package:

```
make
```

Install the package:

```
make install
```

Details on this package are located in Section 10.11.2, “Contents of MPFR.”

5.9. MPC-1.0.3

MPC is a C library for the arithmetic of complex numbers with arbitrarily high precision and correct rounding of the result.

5.9.1. Installation of MPC

Prepare MPC for compilation:

```
LDFLAGS="-Wl,-rpath,/cross-tools/lib" \  
./configure \  
  --prefix=/cross-tools \  
  --disable-static \  
  --with-gmp=/cross-tools \  
  --with-mpfr=/cross-tools
```

Compile the package:

```
make
```

Install the package:

```
make install
```

Details on this package are located in Section 10.12.2, “Contents of MPC.”

5.10. ISL-0.17.1

ISL is a library for manipulating sets and relations of integer points bounded by linear constraints.

5.10.1. Installation of ISL

We will install ISL to enable extra functionality for GCC. It is not strictly required, but GCC can link to it to enable its new loop generation feature called Graphite.

Prepare ISL for compilation:

```
LDFLAGS="-Wl,-rpath,/cross-tools/lib" \  
./configure \  
  --prefix=/cross-tools \  
  --disable-static \  
  --with-gmp-prefix=/cross-tools
```

Compile the package:

```
make
```

Install the package:

```
make install
```

Details on this package are located in Section 10.13.2, “Contents of ISL.”

5.11. Cross Binutils-2.28

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

5.11.1. Installation of Cross Binutils

It is important that Binutils be compiled before Glibc and GCC 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
```

Prepare Binutils for compilation:

```
AR=ar AS=as \
../binutils-2.28/configure \
  --prefix=/cross-tools \
  --host=${CLFS_HOST} \
  --target=${CLFS_TARGET} \
  --with-sysroot=${CLFS} \
  --with-lib-path=/tools/lib \
  --disable-nls \
  --disable-static \
  --disable-multilib \
  --enable-gold=yes \
  --enable-plugins \
  --enable-threads \
  --disable-werror
```

The meaning of the new configure options:

AR=ar AS=as

This prevents Binutils from compiling with `${CLFS_HOST}-ar` and `${CLFS_HOST}-as` as they are provided by this package and therefore not installed yet.

--host=\${CLFS_HOST}

When used with `--target`, this creates a cross-architecture executable which creates files for `${CLFS_TARGET}` but runs on `${CLFS_HOST}`.

--target=\${CLFS_TARGET}

When used with `--host`, this creates a cross-architecture executable that creates files for `${CLFS_TARGET}` but runs on `${CLFS_HOST}`.

--with-sysroot=\${CLFS}

Tells configure to build a linker that uses `${CLFS}` as its root directory for its search paths.

--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.

--disable-nls

This disables internationalization as i18n is not needed for the cross-compile tools.

--disable-multilib

This option disables the building of a multilib capable Binutils.

--disable-werror

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

--enable-gold=yes

This option enables the building of the gold linker.

--enable-plugins

This option enables support for plugins.

--enable-threads

This option enables multi-threaded linking for the gold linker.

Compile the package:

```
make
```

Install the package:

```
make install
```

Details on this package are located in Section 10.17.2, “Contents of Binutils.”

5.12. Cross GCC-7.1.0 - Static

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

5.12.1. Installation of Cross GCC Compiler with Static libgcc and no Threads

Here we will compile GCC, as a cross-compiler that will create executables for our target architecture, statically so that it will not need to look for Glibc's startfiles, which do not yet exist in `/tools`. We will use this cross-compiler, plus the cross-linker we have just installed with Binutils, to compile Glibc. After Glibc is installed into `/tools`, we can rebuild GCC so that it will then be able to build executables that link against the libraries in `/tools`.

Make a couple of essential adjustments to GCC's specs to ensure GCC uses our build environment:

```
patch -Np1 -i ../gcc-7.1.0-specs-1.patch
```

Change the StartFile Spec so that GCC looks in `/tools`:

```
echo -en '\n#undef STANDARD_STARTFILE_PREFIX_1\n#define STANDARD_STARTFILE_PREFIX_1 "/tools"'\n
echo -en '\n#undef STANDARD_STARTFILE_PREFIX_2\n#define STANDARD_STARTFILE_PREFIX_2 "/tools"'\n
```

We will create a dummy `limits.h` so the build will not use the one provided by the host distro:

```
touch /tools/include/limits.h
```

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:

```
AR=ar \
LDFLAGS="-Wl,-rpath,/cross-tools/lib" \
../gcc-7.1.0/configure \
  --prefix=/cross-tools \
  --build=${CLFS_HOST} \
  --host=${CLFS_HOST} \
  --target=${CLFS_TARGET} \
  --with-sysroot=${CLFS} \
  --with-local-prefix=/tools \
  --with-native-system-header-dir=/tools/include \
  --disable-shared \
  --with-mpfr=/cross-tools \
  --with-gmp=/cross-tools \
  --with-isl=/cross-tools \
  --with-mpc=/cross-tools \
  --without-headers \
  --with-newlib \
  --disable-decimal-float \
  --disable-libgomp \
  --disable-libssp \
  --disable-libatomic \
  --disable-libitm \
  --disable-lsanitizer \
  --disable-libquadmath \
  --disable-libvtv \
  --disable-libcilkrts \
  --disable-libstdc++-v3 \
  --disable-threads \
  --disable-multilib \
  --enable-languages=c \
  --with-glibc-version=2.25
```

The meaning of the new configure options:

--build=\${CLFS_HOST}

This specifies the system on which the cross-compiler is being built.

--with-local-prefix=/tools

The purpose of this switch is to remove `/usr/local/include` from **gcc**'s include search path. This is not absolutely essential, however, it helps to minimize the influence of the host system.

--with-native-system-headers-dir=/tools/include

This switch ensures that GCC will search for the system headers in `/tools/include` and that host system headers will not be searched.

--disable-shared

This tells GCC not to create a shared library.

--without-headers

Disables GCC from using the target's Libc when cross compiling.

--with-newlib

This causes GCC to enable the `inhibit_libc` flag, which prevents `libgcc` from building code that uses `libc` support.

--disable-decimal-float

Disables support for the C decimal floating point extension.

*--disable-lib**

These options prevent GCC from building a number of libraries that are not needed at this time.

--disable-threads

This will prevent GCC from looking for the multi-thread include files, since they haven't been created for this architecture yet. GCC will be able to find the multi-thread information after the Glibc headers are created.

--with-system-zlib

This tells GCC to link to the system-installed `zlib` instead of the one in its source tree.

--enable-languages=c

This option ensures that only the C compiler is built.

--with-glibc-version=2.25

Needed when bootstrapping a cross toolchain without the header files available for building the initial bootstrap compiler.

Continue with compiling the package:

```
make all-gcc all-target-libgcc
```

The meaning of the new make options:

all-gcc all-target-libgcc

Compiles only the parts of GCC that are needed at this time, rather than the full package.

Install the package:

```
make install-gcc install-target-libgcc
```

Details on this package are located in Section 10.18.2, “Contents of GCC.”

5.13. Glibc-2.25

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.

5.13.1. Installation of Glibc

It should be noted that compiling Glibc in any way other than the method suggested in this book puts the stability of the system at risk.

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:

```
BUILD_CC="gcc" \
CC="${CLFS_TARGET}-gcc" \
AR="${CLFS_TARGET}-ar" \
RANLIB="${CLFS_TARGET}-ranlib" \
../glibc-2.25/configure \
  --prefix=/tools \
  --host=${CLFS_TARGET} \
  --build=${CLFS_HOST} \
  --enable-kernel=3.12.0 \
  --with-binutils=/cross-tools/bin \
  --with-headers=/tools/include \
  --enable-obsolete-rpc
```

The meaning of the new configure options:

BUILD_CC="gcc"

This sets Glibc to use the current compiler on our system. This is used to create the tools Glibc uses during its build.

CC="\${CLFS_TARGET}-gcc"

This forces Glibc to use the GCC compiler that we made for our target architecture.

AR="\${CLFS_TARGET}-ar"

This forces Glibc to use the **ar** utility we made for our target architecture.

RANLIB="\${CLFS_TARGET}-ranlib"

This forces Glibc to use the **ranlib** utility we made for our target architecture.

--enable-kernel=3.12.0

This tells Glibc to compile the library with support for 3.12.0 and later Linux kernels.

--with-binutils=/cross-tools/bin

This tells Glibc to use the Binutils that are specific to our target architecture.

--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.

`--enable-obsolete-rpc`

This tells Glibc to install rpc headers that are not installed by default but may be needed by other packages.

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. You might also see a similar (also harmless) message about missing **autoconf**.

Compile the package:

```
make
```

Install the package:

```
make install
```

Details on this package are located in Section 10.7.5, “Contents of Glibc.”

5.14. Cross GCC-7.1.0 - Final

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

5.14.1. Installation of GCC Cross Compiler

Make a couple of essential adjustments to GCC's specs to ensure GCC uses our build environment:

```
patch -Np1 -i ../gcc-7.1.0-specs-1.patch
```

Change the StartFile Spec so that GCC looks in /tools:

```
echo -en '\n#undef STANDARD_STARTFILE_PREFIX_1\n#define STANDARD_STARTFILE_PREFIX_1 "/tools" \n' > gcc-7.1.0-specs-1.patch
echo -en '\n#undef STANDARD_STARTFILE_PREFIX_2\n#define STANDARD_STARTFILE_PREFIX_2 "/tools" \n' > gcc-7.1.0-specs-1.patch
```

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:

```
AR=ar \
LDLAGS="-Wl,-rpath,/cross-tools/lib" \
../gcc-7.1.0/configure \
  --prefix=/cross-tools \
  --build=${CLFS_HOST} \
  --target=${CLFS_TARGET} \
  --host=${CLFS_HOST} \
  --with-sysroot=${CLFS} \
  --with-local-prefix=/tools \
  --with-native-system-header-dir=/tools/include \
  --disable-nls \
  --disable-static \
  --enable-languages=c,c++ \
  --disable-multilib \
  --with-mpc=/cross-tools \
  --with-mpfr=/cross-tools \
  --with-gmp=/cross-tools \
  --with-isl=/cross-tools
```

The meaning of the new configure options:

--enable-languages=c,c++

This option ensures that only the C and C++ compilers are built.

Continue with compiling the package:

```
make AS_FOR_TARGET="${CLFS_TARGET}-as" \
  LD_FOR_TARGET="${CLFS_TARGET}-ld"
```

Install the package:

```
make install
```


Details on this package are located in Section 10.18.2, “Contents of GCC.”

Part IV. Building the Basic Tools

Chapter 6. Constructing a Temporary System

6.1. Introduction

This chapter shows how to compile and install a minimal Linux system. This system will contain just enough tools to start constructing the final CLFS system in Installing Basic System Software and allow a working environment with more user convenience than a minimum environment would.

The tools in this chapter are cross-compiled using the toolchain in `/cross-tools` and will be installed under the `${CLFS}/tools` directory to keep them separate from the files installed in Installing Basic System Software and the host production directories. Since the packages compiled here are temporary, we do not want them to pollute the soon-to-be CLFS system.

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

```
echo ${CLFS}
```

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

During this section of the build you will see several WARNING messages like the ones below. It is safe to ignore these messages.

```
configure: WARNING: result yes guessed because of cross compilation
configure: WARNING: cannot check WCONTINUED if cross compiling -- defaulting to no
```

6.2. Build Variables

Setup target-specific variables for the compiler and linkers:

```
export CC="${CLFS_TARGET}-gcc"
export CXX="${CLFS_TARGET}-g++"
export AR="${CLFS_TARGET}-ar"
export AS="${CLFS_TARGET}-as"
export RANLIB="${CLFS_TARGET}-ranlib"
export LD="${CLFS_TARGET}-ld"
export STRIP="${CLFS_TARGET}-strip"
```

Then add the build variables to `~/ .bashrc` to prevent issues if you stop and come back later:

```
echo export CC=\"\"${CC}\"\" >> ~/.bashrc
echo export CXX=\"\"${CXX}\"\" >> ~/.bashrc
echo export AR=\"\"${AR}\"\" >> ~/.bashrc
echo export AS=\"\"${AS}\"\" >> ~/.bashrc
echo export RANLIB=\"\"${RANLIB}\"\" >> ~/.bashrc
echo export LD=\"\"${LD}\"\" >> ~/.bashrc
echo export STRIP=\"\"${STRIP}\"\" >> ~/.bashrc
```

6.3. GMP-6.1.2

GMP is a library for arithmetic on arbitrary precision integers, rational numbers, and floating-point numbers.

6.3.1. Installation of GMP

As with the Cross-Tools, we will compile GMP, MPFR, MPC, and ISL so that GCC can use them, though this time we will cross-compile them into `/tools`.

Prepare GMP for compilation:

```
CC_FOR_BUILD=gcc \
./configure \
  --prefix=/tools \
  --build=${CLFS_HOST} \
  --host=${CLFS_TARGET} \
  --enable-cxx
```

The meaning of the new configure option:

`CC_FOR_BUILD=gcc`

Tells **configure** to use the host's **gcc** instead of our cross-compiler to build native tools it needs while compiling.

Compile the package:

```
make
```

Install the package:

```
make install
```

Details on this package are located in Section 10.10.2, “Contents of GMP.”

6.4. MPFR-3.1.5

The MPFR library is a C library for multiple-precision floating-point computations with correct rounding.

6.4.1. Installation of MPFR

Apply a patch with upstream fixes:

```
patch -Np1 -i ../mpfr-3.1.5-fixes-1.patch
```

Prepare MPFR for compilation:

```
./configure \
  --prefix=/tools \
  --build=${CLFS_HOST} \
  --host=${CLFS_TARGET}
```

Compile the package:

```
make
```

Install the package:

```
make install
```

Details on this package are located in Section 10.11.2, “Contents of MPFR.”

6.5. MPC-1.0.3

MPC is a C library for the arithmetic of complex numbers with arbitrarily high precision and correct rounding of the result.

6.5.1. Installation of MPC

Prepare MPC for compilation:

```
./configure \
  --prefix=/tools \
  --build=${CLFS_HOST} \
  --host=${CLFS_TARGET}
```

Compile the package:

```
make
```

Install the package:

```
make install
```

Details on this package are located in Section 10.12.2, “Contents of MPC.”

6.6. ISL-0.17.1

ISL is a library for manipulating sets and relations of integer points bounded by linear constraints.

6.6.1. Installation of ISL

Prepare ISL for compilation:

```
./configure \  
  --prefix=/tools \  
  --build=${CLFS_HOST} \  
  --host=${CLFS_TARGET}
```

Compile the package:

```
make
```

Install the package:

```
make install
```

Details on this package are located in Section 10.13.2, “Contents of ISL.”

6.7. Zlib-1.2.11

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

6.7.1. Installation of Zlib

Several packages in the temporary system use Zlib, including Binutils, GCC, and Util-linux, so we will add it to `/tools`.

Prepare Zlib for compilation:

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

Compile the package:

```
make
```

Install the package:

```
make install
```

Details on this package are located in Section 10.14.2, “Contents of Zlib.”

6.8. Binutils-2.28

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

6.8.1. Installation of Binutils

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-2.28/configure \
  --prefix=/tools \
  --build=${CLFS_HOST} \
  --host=${CLFS_TARGET} \
  --target=${CLFS_TARGET} \
  --with-lib-path=/tools/lib \
  --disable-nls \
  --enable-shared \
  --disable-multilib \
  --enable-gold=yes \
  --enable-plugins \
  --with-system-zlib \
  --enable-threads
```

The meaning of the new configure option:

--enable-shared

When this is specified, Binutils will create a shared `libbfd` and link its programs to it.

Compile the package:

```
make
```

Install the package:

```
make install
```

Details on this package are located in Section 10.17.2, “Contents of Binutils.”

6.9. GCC-7.1.0

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

6.9.1. Installation of GCC

Make a couple of essential adjustments to GCC's specs to ensure GCC uses our build environment:

```
patch -Np1 -i ../gcc-7.1.0-specs-1.patch
```

Change the StartFile Spec so that GCC looks in /tools:

```
echo -en '\n#undef STANDARD_STARTFILE_PREFIX_1\n#define STANDARD_STARTFILE_PREFIX_1 "/tools" \n' > gcc/specs
echo -en '\n#undef STANDARD_STARTFILE_PREFIX_2\n#define STANDARD_STARTFILE_PREFIX_2 "/tools" \n' > gcc/specs
```

Apply a **sed** substitution that will suppress the execution of the **fixincludes** script:

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

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

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

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

Prepare GCC for compilation:

```
../gcc-7.1.0/configure \
  --prefix=/tools \
  --build=${CLFS_HOST} \
  --host=${CLFS_TARGET} \
  --target=${CLFS_TARGET} \
  --with-local-prefix=/tools \
  --disable-multilib \
  --enable-languages=c,c++ \
  --with-system-zlib \
  --with-native-system-header-dir=/tools/include \
  --disable-libssp \
  --enable-install-libiberty
```

The meaning of the new configure option:

--enable-install-libiberty

Allows GCC to build and install `libiberty.a` and its associated headers. This library is needed for some packages to compile.

Compile the package:

```
make AS_FOR_TARGET="${AS}" \
  LD_FOR_TARGET="${LD}"
```

Install the package:

```
make install
```

Details on this package are located in Section 10.18.2, “Contents of GCC.”

6.10. Ncurses-6.0

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

6.10.1. Installation of Ncurses

We will need Ncurses for several other packages in the temporary environment, including Bash, Util-linux, and Vim.

Prepare Ncurses for compilation:

```
./configure \
  --prefix=/tools \
  --with-shared \
  --build=${CLFS_HOST} \
  --host=${CLFS_TARGET} \
  --without-debug \
  --without-ada \
  --enable-overwrite \
  --with-build-cc=gcc
```

The meaning of the new configure options:

--with-shared

This tells Ncurses to create a shared library.

--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 when building the final system.

--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.

--with-build-cc=gcc

This tells Ncurses which compiler to use to build native tools when cross-compiling.

Compile the package:

```
make
```

Install the package:

```
make install
```

Details on this package are located in Section 10.24.2, “Contents of Ncurses.”

6.11. Bash-4.4

The Bash package contains the Bourne-Again SHell.

6.11.1. Installation of Bash

The following patch contains updates from the maintainer. The maintainer of Bash only releases these patches to fix serious issues:

```
patch -Np1 -i ../bash-4.4-branch_update-1.patch
```

When Bash is cross-compiled, it cannot test for the presence of named pipes, among other things. If you used **su** to become an unprivileged user, this combination will cause Bash to build without *process substitution*, which will break one of the C++ test scripts in `glibc`. The following prevents future problems by skipping the check for named pipes, as well as other tests that can not run while cross-compiling or that do not run properly:

```
cat > config.cache << "EOF"
ac_cv_func_mmap_fixed_mapped=yes
ac_cv_func_strcoll_works=yes
ac_cv_func_working_mktime=yes
bash_cv_func_sigsetjmp=present
bash_cv_getcwd_malloc=yes
bash_cv_job_control_missing=present
bash_cv_printf_a_format=yes
bash_cv_sys_named_pipes=present
bash_cv_ulimit_maxfds=yes
bash_cv_under_sys_siglist=yes
bash_cv_unusable_rtsigs=no
gt_cv_int_divbyzero_sigfpe=yes
EOF
```

Prepare Bash for compilation:

```
./configure \
  --prefix=/tools \
  --build=${CLFS_HOST} \
  --host=${CLFS_TARGET} \
  --without-bash-malloc \
  --cache-file=config.cache
```

The meaning of the new configure option:

`--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
```

Install the package:

```
make install
```

Details on this package are located in Section 10.39.2, “Contents of Bash.”

6.12. 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**.

6.12.1. Installation of Bzip2

Bzip2's default `Makefile` target automatically runs the test suite as well. Disable the tests since they won't work on a multi-architecture build:

```
cp -v Makefile{,.orig}
sed -e '/^all/s/ test$//' Makefile.orig > Makefile
```

The Bzip2 package does not contain a **configure** script. Compile it with:

```
make CC="${CC}" AR="${AR}" RANLIB="${RANLIB}"
```

Install the package:

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

Details on this package are located in Section 10.33.2, “Contents of Bzip2.”

6.13. Check-0.11.0

The Check package is a unit testing framework for C.

6.13.1. Installation of Check

We will install Check into `/tools` to satisfy a dependency on it for Kbd and Libpipeline in the final system.

Prepare Check for compilation:

```
./configure \
  --prefix=/tools \
  --build=${CLFS_HOST} \
  --host=${CLFS_TARGET}
```

Build the package:

```
make
```

Install the package:

```
make install
```

6.13.2. 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 C the Check unit testing framework
libcheck.{a,so}	Contains functions that allow Check to be called from a test program

6.14. Coreutils-8.27

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

6.14.1. Installation of Coreutils

Prepare Coreutils for compilation:

```
./configure \
  --prefix=/tools \
  --build=${CLFS_HOST} \
  --host=${CLFS_TARGET} \
  --enable-install-program=hostname \
  --cache-file=config.cache
```

The meaning of the new configure option:

--enable-install-program=hostname

Tells Coreutils to install **hostname**, which is needed for the Perl test suite.

Adjust the Makefile so man pages aren't generated:

```
sed -i -e 's/^man1_MANS/#man1_MANS/' Makefile
```

Compile the package:

```
make
```

Install the package:

```
make install
```

Details on this package are located in Section 10.29.2, “Contents of Coreutils.”

6.15. Diffutils-3.6

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

6.15.1. Installation of Diffutils

Prepare Diffutils for compilation:

```
./configure \  
  --prefix=/tools \  
  --build=${CLFS_HOST} \  
  --host=${CLFS_TARGET}
```

Compile the package:

```
make
```

Install the package:

```
make install
```

Details on this package are located in Section 10.41.2, “Contents of Diffutils.”

6.16. File-5.31

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

6.16.1. Installation of File

Prepare File for compilation:

```
./configure \  
  --prefix=/tools \  
  --build=${CLFS_HOST} \  
  --host=${CLFS_TARGET}
```

Compile the package:

```
make
```

Install the package:

```
make install
```

Details on this package are located in Section 10.42.2, “Contents of File.”

6.17. Findutils-4.6.0

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).

6.17.1. Installation of Findutils

The following cache entries set the values for tests that do not run while cross-compiling:

```
echo "gl_cv_func_wcwidth_works=yes" > config.cache
echo "ac_cv_func_fnmatch_gnu=yes" >> config.cache
```

Prepare Findutils for compilation:

```
./configure \
  --prefix=/tools \
  --build=${CLFS_HOST} \
  --host=${CLFS_TARGET} \
  --cache-file=config.cache
```

Compile the package:

```
make
```

Install the package:

```
make install
```

Details on this package are located in Section 10.44.2, “Contents of Findutils.”

6.18. Gawk-4.1.4

The Gawk package contains programs for manipulating text files.

6.18.1. Installation of Gawk

Prepare Gawk for compilation:

```
./configure \  
  --prefix=/tools \  
  --build=${CLFS_HOST} \  
  --host=${CLFS_TARGET}
```

Compile the package:

```
make
```

Install the package:

```
make install
```

Details on this package are located in Section 10.43.2, “Contents of Gawk.”

6.19. Gettext-0.19.8.1

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

6.19.1. Installation of Gettext

Many packages' installation procedures use the **msgfmt** program for i18n support, so we will compile and install it into `/tools`. Attr also needs **msgmerge** and **xgettext**, so we will install those as well.

Only the programs in the `gettext-tools` directory need to be installed for the temp-system:

```
cd gettext-tools
```

Prepare Gettext for compilation:

```
EMACS="no" \
./configure \
  --prefix=/tools \
  --build=${CLFS_HOST} \
  --host=${CLFS_TARGET} \
  --disable-shared
```

The meaning of the new configure options:

EMACS="no"

Prevents the configure script from installing Emacs Lisp files as the test is known to hang on some hosts.

Compile the required programs and support library:

```
make -C gnulib-lib
make -C intl pluralx.c
make -C src msgfmt msgmerge xgettext
```

Install the **msgfmt**, **msgmerge** and **xgettext** binaries:

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

Details on this package are located in Section 10.45.2, “Contents of Gettext.”

6.20. Grep-3.0

The Grep package contains programs for searching through files.

6.20.1. Installation of Grep

Prepare Grep for compilation:

```
./configure \  
  --prefix=/tools \  
  --build=${CLFS_HOST} \  
  --host=${CLFS_TARGET} \  
  --without-included-regex
```

The meaning of the new configure option:

--without-included-regex

When cross-compiling, Grep's **configure** assumes there is no usable `regex.h` installed and instead uses the one included with Grep. This switch forces the use of the regex functions from Glibc.

Compile the package:

```
make
```

Install the package:

```
make install
```

Details on this package are located in Section 10.47.2, “Contents of Grep.”

6.21. Gzip-1.8

The Gzip package contains programs for compressing and decompressing files.

6.21.1. Installation of Gzip

Prepare Gzip for compilation:

```
./configure \  
  --prefix=/tools \  
  --build=${CLFS_HOST} \  
  --host=${CLFS_TARGET}
```

Compile the package:

```
make
```

Install the package:

```
make install
```

Details on this package are located in Section 10.50.2, “Contents of Gzip.”

6.22. Make-4.2.1

The Make package contains a program for compiling packages.

6.22.1. Installation of Make

Prepare Make for compilation:

```
./configure \  
  --prefix=/tools \  
  --build=${CLFS_HOST} \  
  --host=${CLFS_TARGET}
```

Compile the package:

```
make
```

Install the package:

```
make install
```

Details on this package are located in Section 10.55.2, “Contents of Make.”

6.23. Patch-2.7.5

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

6.23.1. Installation of Patch

Prepare Patch for compilation:

```
./configure \
  --prefix=/tools \
  --build=${CLFS_HOST} \
  --host=${CLFS_TARGET}
```

Compile the package:

```
make
```

Install the package:

```
make install
```

Details on this package are located in Section 10.61.2, “Contents of Patch.”

6.24. Sed-4.4

The Sed package contains a stream editor.

6.24.1. Installation of Sed

Prepare Sed for compilation:

```
./configure \  
  --prefix=/tools \  
  --build=${CLFS_HOST} \  
  --host=${CLFS_TARGET}
```

Compile the package:

```
make
```

Install the package:

```
make install
```

Details on this package are located in Section 10.22.2, “Contents of Sed.”

6.25. Tar-1.29

The Tar package contains an archiving program.

6.25.1. Installation of Tar

Configure can not properly determine the results of a few tests. Set them manually:

```
cat > config.cache << EOF
gl_cv_func_wcwidth_works=yes
gl_cv_func_btowc_eof=yes
ac_cv_func_malloc_0_nonnull=yes
gl_cv_func_mbrtowc_incomplete_state=yes
gl_cv_func_mbrtowc_nul_retval=yes
gl_cv_func_mbrtowc_null_arg1=yes
gl_cv_func_mbrtowc_null_arg2=yes
gl_cv_func_mbrtowc_retval=yes
gl_cv_func_wcrtomb_retval=yes
EOF
```

Prepare Tar for compilation:

```
./configure \
  --prefix=/tools \
  --build=${CLFS_HOST} \
  --host=${CLFS_TARGET} \
  --cache-file=config.cache
```

Compile the package:

```
make
```

Install the package:

```
make install
```

Details on this package are located in Section 10.65.2, “Contents of Tar.”

6.26. Texinfo-6.3

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

6.26.1. Installation of Texinfo

Prepare Texinfo for compilation:

```
PERL=/usr/bin/perl \  
./configure \  
  --prefix=/tools \  
  --build=${CLFS_HOST} \  
  --host=${CLFS_TARGET}
```

The meaning of the new configure option:

PERL=/usr/bin/perl

This forces Texinfo to use */usr/bin* as the location of **perl**, as some host systems may have it in */bin*.

Compile the package:

```
make
```

Install the package:

```
make install
```

Details on this package are located in Section 10.66.2, “Contents of Texinfo.”

6.27. Util-linux-2.29.2

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

6.27.1. Installation of Util-linux

Prepare Util-linux for compilation:

```
./configure \
  --prefix=/tools \
  --build=${CLFS_HOST} \
  --host=${CLFS_TARGET} \
  --disable-makeinstall-chown \
  --disable-makeinstall-setuid \
  --disable-nologin \
  --without-python
```

The meaning of the new configure option:

--disable-makeinstall-chown

This prevents Util-linux from trying to perform any `chown` commands when it is installed.

--disable-makeinstall-setuid

This prevents Util-linux from enabling the `setuid` bit on any of its installed programs.

--disable-nologin

This prevents Util-linux from installing **nologin**.

--without-python

This avoids the building of unneeded Python bindings when Python is installed on the host system.

Adjust the Makefile

```
sed -i 's/-lnursesw -ltinfo/-lnurses/' Makefile
sed -i 's/LIBNCURSESW/LIBNCURSES/' config.h
```

Compile the package:

```
make
```

Install the package:

```
make install
```

Details on this package are located in Section 10.67.3, “Contents of Util-linux.”

6.28. Vim-8.0

The Vim package contains a powerful text editor.

6.28.1. Installation of VIM

We will cross-compile Vim so that we can have a text editor in `/tools`. Vim is not technically necessary in the temporary system, in that it is not there to satisfy any package dependencies in the final system, but we believe that a text editor is an extremely useful tool to have there.

The following patch merges all updates from the 8.0 Branch from the Vim developers:

```
patch -Np1 -i ../vim-8.0-branch_update-1.patch
```

The **configure** script is full of logic that aborts at the first sign of cross compiling. Work around this by setting the cached values of several tests with the following command:

```
cat > src/auto/config.cache << "EOF"
vim_cv_getcwd_broken=no
vim_cv_memmove_handles_overlap=yes
vim_cv_stat_ignores_slash=no
vim_cv_terminfo=yes
vim_cv_toupper_broken=no
vim_cv_tty_group=world
vim_cv_tgent=zero
EOF
```

Change the default location of the `vimrc` configuration file to `/tools/etc`:

```
echo '#define SYS_VIMRC_FILE "/tools/etc/vimrc"' >> src/feature.h
```

Prepare Vim for compilation:

```
./configure \
  --build=${CLFS_HOST} \
  --host=${CLFS_TARGET} \
  --prefix=/tools \
  --enable-gui=no \
  --disable-gtktest \
  --disable-xim \
  --disable-gpm \
  --without-x \
  --disable-netbeans \
  --with-tlib=ncurses
```

The meaning of the new configure options:

```
--enable-gui=no --disable-gtktest --disable-xim --disable-gpm --without-x --
disable-netbeans
```

These options prevent Vim from trying to link to libraries that might be on the host but won't exist inside the temporary build environment.

```
--with-tlib=ncurses
```

Tells Vim to use Ncurses as its terminal library.

Compile the package:

```
make
```

Install the package:

```
make -j1 install
```

Many users are accustomed to using **vi** instead of **vim**. Some programs, such as **vigr** and **vipw**, also use **vi**. Create a symlink to permit execution of **vim** when users habitually enter **vi** and allow programs that use **vi** to work:

```
ln -sv vim /tools/bin/vi
```

Create a temporary vimrc to make it function more the way you may expect it to. This is explained more in the final system:

```
cat > /tools/etc/vimrc << "EOF"
" Begin /tools/etc/vimrc

set nocompatible
set backspace=2
set ruler
syntax on

" End /tools/etc/vimrc
EOF
```

Details on this package are located in Section 10.68.3, “Contents of Vim.”

6.29. XZ Utils-5.2.3

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

6.29.1. Installation of XZ Utils

Prepare XZ Utils for compilation:

```
./configure \
  --prefix=/tools \
  --build=${CLFS_HOST} \
  --host=${CLFS_TARGET}
```

Compile the package:

```
make
```

Install the package:

```
make install
```

Details on this package are located in Section 10.56.2, “Contents of XZ Utils.”

6.30. To Boot or to Chroot?

There are two different ways you can proceed from this point to build the final system. You can build a kernel, a bootloader, and a few other utilities, boot into the temporary system, and build the rest there. Alternatively, you can mount a few virtual filesystems and chroot into the temporary system.

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 CLFS partition. This is very similar to rebooting and instructing the kernel to mount the CLFS partition as the root partition. The major advantage is that “chrooting” allows the builder to continue using the host while CLFS is being built. While waiting for package compilation to complete, a user can switch to a different virtual console (VC) or X desktop and continue using the computer as normal.

The main downside to chrooting is that you are more limited in when you can use it - booting will always work for any CLFS build, but the chroot method can only be used when you are building on the same architecture. For example, if you are building on, and for, an x86 system, you can simply chroot. Booting is required when you are compiling for a different architecture, such as building a PowerPC system from an x86. The rule of thumb here is that if the architectures match and you are running the same series kernel (specifically, a 3.12.0 or newer Linux kernel) you can just chroot. If you aren't running the same series kernel, or are wanting to run a different ABI, you will need to use the boot option.

If you are in any doubt about this, you can try the following commands to see if you can chroot:

```
/tools/lib/libc.so.6  
/tools/bin/gcc -v
```

If either of these commands fail, you will have to follow the boot method.

For the boot method, follow [If You Are Going to Boot](#).

For the chroot method, follow [If You Are Going to Chroot](#).

Chapter 7. If You Are Going to Boot

7.1. Introduction

This chapter shows how to complete the build of temporary tools to create a minimal system that will be used to boot the target machine and to build the final system packages.

7.2. Bc-1.07.1

The Bc package contains an arbitrary precision numeric processing language.

7.2.1. Installation of Bc

We will install a **bc** program that can run on the host system, as this is needed to compile the kernel.

Prepare Bc for compilation:

```
CC=gcc \  
./configure \  
--prefix=/cross-tools
```

The meaning of the configure option:

CC=gcc

This ensures that we use the host's compiler to build Bc, since we need it to run on the host system.

Compile the package:

```
make
```

Install the package:

```
make install
```

Details on this package are located in Section 10.40.2, “Contents of Bc.”

7.3. Boot-scripts for CLFS 3.0-20140710

The Boot-scripts package contains a set of scripts to start/stop the CLFS system at bootup/shutdown.

7.3.1. Installation of Boot-scripts

Install the package:

```
make DESTDIR=/tools install-boot-scripts
```

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 `/tools/etc/sysconfig/clock` 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.

If you do not know whether or not the hardware clock is set to UTC, you can find out after you have booted the new machine by running the **hwclock --localtime --show** command, and if necessary editing the `/tools/etc/sysconfig/clock` file. The worst that will happen if you make a wrong guess here is that the time displayed will be wrong.

Change the value of the UTC variable below to a value of 0 (zero) if the hardware clock is *not* set to UTC time.

```
cat > /tools/etc/sysconfig/clock << "EOF"
# Begin /tools/etc/sysconfig/clock

UTC=1

# End /tools/etc/sysconfig/clock
EOF
```

7.3.2. Contents of Boot-scripts

Installed scripts: checkfs, cleanfs, functions, halt, localnet, mountfs, mountkernfs, rc, reboot, sendsignals, setclock, swap, and udev.

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 <code>/run/</code> and <code>/var/lock/</code> ; it re-creates <code>/run/utmp</code> and removes the possibly present <code>/etc/nologin</code> , <code>/fastboot</code> , and <code>/forcefsck</code> files
functions	Contains common functions, such as error and status checking, that are used by several boot-scripts
halt	Halts the system
localnet	Sets up the system's hostname and local loopback device
mountfs	Mounts all file systems, except ones that are marked <i>noauto</i> or are network based
mountkernfs	Mounts virtual kernel file systems, such as <code>proc</code>

rc	The master run-level control script; it is responsible for running all the other boot-scripts 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
swap	Enables and disables swap files and partitions
udev	Starts and stops the Eudev daemon

7.4. E2fsprogs-1.43.4

The E2fsprogs package contains the utilities for handling the `ext2` file system. It also supports the `ext3` and `ext4` journaling file systems.

7.4.1. 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=/tools \
  --enable-elf-shlibs \
  --build=${CLFS_HOST} \
  --host=${CLFS_TARGET} \
  --disable-libblkid \
  --disable-libuuid \
  --disable-fsck \
  --disable-uuid
```

The meaning of the configure options:

--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 `uuid` daemon, and the `fsck` wrapper, as Util-Linux installed all of them earlier.

Compile the package:

```
make
```

Install the binaries, documentation and shared libraries:

```
make install
```

Install the static libraries and headers:

```
make install-libs
```

Details on this package are located in Section 10.28.2, “Contents of E2fsprogs.”

7.5. Kmod-24

The Kmod package contains programs for loading, inserting and removing kernel modules for Linux. Kmod replaces the Module-Init-tools package.

7.5.1. Installation of Kmod

The following **sed** changes Kmod's default module search location to `/tools/lib/modules`:

```
cp -v libkmod/libkmod.c{,.orig}
sed '/dirname_default_prefix /s@/lib/modules@/tools&@' \
    libkmod/libkmod.c.orig > libkmod/libkmod.c
```

Prepare Kmod for compilation:

```
./configure \
    --prefix=/tools \
    --build=${CLFS_HOST} \
    --host=${CLFS_TARGET} \
    --with-xz \
    --with-zlib
```

The meaning of the new configure options:

`--with-zlib` `--with-xz`

These allow the Kmod package to handle zlib and XZ compressed kernel modules.

Compile the package:

```
make
```

Install the package:

```
make install
```

Create symbolic links for programs that expect Module-Init-Tools:

```
ln -sfv kmod /tools/bin/lsmmod
for tool in depmod insmod modprobe modinfo rmmmod; do
    ln -sv ../bin/kmod /tools/sbin/${tool}
done
```

Details on this package are located in Section 10.60.2, “Contents of Kmod.”

7.6. Shadow-4.5

The Shadow package contains programs for handling passwords in a secure way.

7.6.1. Installation of Shadow

Disable the installation of the **groups** program and man pages, as better versions of these programs are provided by Coreutils, Util-linux and Man-pages. Also, prevent Shadow from setting the suid bit on its installed programs:

```
cp -v src/Makefile.in{,.orig}
sed -e 's/groups$(EXEEXT) //' \
    -e 's/\\(^suidu*bins = \\).*/\\1\\/' \
    src/Makefile.in.orig > src/Makefile.in
```

Tell Shadow to use **passwd** in `/tools/bin`:

```
cat > config.cache << "EOF"
shadow_cv_passwd_dir=/tools/bin
EOF
```

Prepare Shadow for compilation:

```
./configure \
    --prefix=/tools \
    --build=${CLFS_HOST} \
    --host=${CLFS_TARGET} \
    --cache-file=config.cache
```

Compile the package:

```
make
```

Install the package:

```
make install
```

Details on this package are located in Section 10.25.4, “Contents of Shadow.”

7.7. Sysvinit-2.88dsf

The Sysvinit package contains programs for controlling the startup, running, and shutdown of the system.

7.7.1. Installation of Sysvinit

Apply a patch to prevent installation of unneeded programs, and allow Sysvinit to be installed in `/tools`:

```
patch -Np1 -i ../sysvinit-2.88dsf-tools_updates-1.patch
```

Compile the package:

```
make -C src clobber
make -C src CC="${CC}"
```

Install the package:

```
make -C src ROOT=/tools install
```

7.7.2. Configuring Sysvinit

Create a new file `/tools/etc/inittab` by running the following:

```
cat > /tools/etc/inittab << "EOF"
# Begin /tools/etc/inittab

id:3:initdefault:

si::sysinit:/tools/etc/rc.d/init.d/rc sysinit

10:0:wait:/tools/etc/rc.d/init.d/rc 0
11:S1:wait:/tools/etc/rc.d/init.d/rc 1
12:2:wait:/tools/etc/rc.d/init.d/rc 2
13:3:wait:/tools/etc/rc.d/init.d/rc 3
14:4:wait:/tools/etc/rc.d/init.d/rc 4
15:5:wait:/tools/etc/rc.d/init.d/rc 5
16:6:wait:/tools/etc/rc.d/init.d/rc 6

ca:12345:ctrlaltdel:/tools/sbin/shutdown -t1 -a -r now

su:S016:once:/tools/sbin/sulogin

EOF
```

The following command adds the standard virtual terminals to `/tools/etc/inittab`. If your system only has a serial console skip the following command:

```
cat >> /tools/etc/inittab << "EOF"
1:2345:respawn:/tools/sbin/agetty --noclear -I '\033(K' tty1 9600
2:2345:respawn:/tools/sbin/agetty --noclear -I '\033(K' tty2 9600
3:2345:respawn:/tools/sbin/agetty --noclear -I '\033(K' tty3 9600
4:2345:respawn:/tools/sbin/agetty --noclear -I '\033(K' tty4 9600
5:2345:respawn:/tools/sbin/agetty --noclear -I '\033(K' tty5 9600
6:2345:respawn:/tools/sbin/agetty --noclear -I '\033(K' tty6 9600

EOF
```

If your system has a serial console, run the following command to add the entry to `/tools/etc/inittab`.

```
cat >> /tools/etc/inittab << "EOF"
c0:12345:respawn:/tools/sbin/agetty --noclear 115200 ttyS0 vt100

EOF
```

Finally, add the end line to `/tools/etc/inittab`.

```
cat >> /tools/etc/inittab << "EOF"
# End /tools/etc/inittab
EOF
```

The `-I '\033(K'` option tells **agetty** to send this escape sequence to the terminal before doing anything else. This escape sequence switches the console character set to a user-defined one, which can be modified by running the **setfont** program. Sending this escape sequence is necessary for people who use non-ISO 8859-1 screen fonts, but it does not affect native English speakers.

7.7.3. Contents of Sysvinit

Installed programs: bootlogd, fstab-decode, halt, init, killall5, poweroff (link to halt), reboot (link to halt), runlevel, shutdown, and telinit (link to init)

Short Descriptions

bootlogd	Logs boot messages to a log file
fstab-decode	Runs a command with fstab-encoded arguments
halt	Normally invokes shutdown with the <code>-h</code> option, except when already in run-level 0, then it tells the kernel to halt the system; it notes in the file <code>/var/log/wtmp</code> 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 processes 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
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 <code>/run/utmp</code>
shutdown	Brings the system down in a secure way, signaling all processes and notifying all logged-in users
telinit	Tells init which run-level to change to

7.8. Eudev-1.7

The Eudev package contains programs for dynamic creation of device nodes.

7.8.1. Installation of Eudev

Prepare Eudev for compilation:

```
./configure \
  --prefix=/tools \
  --build=${CLFS_HOST} \
  --host=${CLFS_TARGET} \
  --disable-introspection \
  --disable-gtk-doc-html \
  --disable-gudev \
  --disable-keymap \
  --with-firmware-path=/tools/lib/firmware \
  --enable-libkmod
```

The meaning of the new configure options:

--disable-introspection --disable-gtk-doc-html --disable-gudev --disable-keymap

These switches disable several features which are not needed for the temporary system and have additional dependencies.

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

This allows Eudev to load firmware from `/tools/lib/firmware` instead of the default location of `/lib/firmware`.

--enable-libkmod

Allows Eudev to load modules by using `libkmod` directly.

Compile the package:

```
make
```

Install the package:

```
make install
```

Create a directory for storing firmware that can be loaded by **udev**:

```
install -dv /tools/lib/firmware
```

Create a dummy rule so that Eudev will name ethernet devices properly for the system.

```
echo "# dummy, so that network is once again on eth*" > \
/tools/etc/udev/rules.d/80-net-name-slot.rules
```

7.8.2. Contents of Eudev

Installed programs:	ata_id, cdrom_id, collect, create_floppy_devices, edd_id, firmware.sh, fstab_import, path_id, scsi_id, udevadm, udevd, usb_id, v4l_id, write_cd_rules, write_net_rules
Installed library:	libudev
Installed directories:	/tools/etc/udev, /tools/lib/firmware, /tools/lib/udev

Short Descriptions

udevadm	Controls the runtime behavior of Eudev, requests kernel events, manages the event queue, and provides simple debugging.
udev	A daemon that reorders hotplug events before submitting them to udev , thus avoiding various race conditions
ata_id	Provides Eudev with a unique string and additional information (uuid, label) for an ATA drive
cdrom_id	Prints the capabilities of a CDROM or DVDROM drive.
collect	Given an ID for the current uevent and a list of IDs (for all target uevents), registers the current ID and indicates whether all target IDs have been registered.
create_floppy_devices	Creates all possible floppy devices based on the CMOS type
edd_id	Identifies x86 disk drives from Enhanced Disk Drive calls
firmware.sh	Script to load firmware for a device
fstab_import	Finds an entry in <code>/etc/fstab</code> that matches the current device, and provides its information to Udev.
path_id	Provides the shortest possible unique hardware path to a device
scsi_id	Retrieves or generates a unique SCSI identifier.
usb_id	Identifies a USB block device.
v4l_id	Determines V4L capabilities for a given device.
write_cd_rules	A script which generates Eudev rules to provide stable names for network interfaces.
write_net_rules	A script which generates Eudev rules to provide stable names for network interfaces.
libudev	A library interface to eudev device information.
<code>/etc/udev</code>	Contains udev configuration files, device permissions, and rules for device naming
<code>/lib/udev</code>	Contains udev helper programs and static devices which get copied to <code>/dev</code> when booted.

7.9. Linux-4.9.21

The Linux package contains the Linux kernel.

7.9.1. Installation of the kernel

Warning

Here a temporary cross-compiled kernel will be built. When configuring it, select the minimal amount of options required to boot the target machine and build the final system. I.e., no support for sound, printers, etc. will be needed.

Also, try to avoid the use of modules if possible, and don't use the resulting kernel image for production systems.

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.

Apply the latest Linux sublevel patch:

```
xzcat ../patch-4.9.21.xz | patch -Np1 -i -
```

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:

```
make ARCH=mips CROSS_COMPILE=${CLFS_TARGET}- menuconfig
```

Compile the kernel image and modules:

```
make ARCH=mips CROSS_COMPILE=${CLFS_TARGET}-
```

If the use of kernel modules can't be avoided, a file in `/etc/modprobe.d` may be needed. Information pertaining to modules and kernel configuration is located in the kernel documentation in the `Documentation` directory of the kernel sources tree. The `modprobe.d` man page may also be of interest.

Install the modules, if the kernel configuration uses them:

```
make ARCH=mips CROSS_COMPILE=${CLFS_TARGET}- \
    INSTALL_MOD_PATH=/tools modules_install
```

Install the firmware, if the kernel configuration uses them:

```
make ARCH=mips CROSS_COMPILE=${CLFS_TARGET}- \
    INSTALL_MOD_PATH=/tools firmware_install
```

After kernel compilation is complete, additional steps are required to complete the installation. Some files need to be copied to the `/tools/boot` directory.

Issue the following commands to install the kernel:

```
mkdir -pv /tools/boot  
cp -v vmlinux /tools/boot/vmlinux-4.9.21  
gzip -9 /tools/boot/vmlinux-4.9.21
```

`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. Issue the following command to install the map file:

```
cp -v System.map /tools/boot/System.map-4.9.21
```

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 /tools/boot/config-4.9.21
```

Details on this package are located in Section 13.2.2, “Contents of Linux.”

7.10. Bootloaders

On MIPS based platforms, we have 2 different bootloaders - Colo for the Cobalt based MIPS machines and Arcload for the SGI machines. At this time, in the boot scenario, the only bootloader we can build and that is usable is Cobalt bootloader. On SGI machines that follow this build method, we recommend to do a netboot. Information about netbooting can be found at the link below.

<http://hints.clfs.org/index.php/Category:Netboot>

7.11. Colo-1.22

The Colo package contains the Cobalt Boot Loader.

7.11.1. Installation of Colo

Note

This bootloader is for the MIPS based cobalt servers RaQ, RaQ2, Qube, or the Qube2.

This patch fixes a relocation error when linking with Binutils:

```
patch -Np1 -i ../colo-1.22-relocation_fix-1.patch
```

Disable -Werror in one of the Makefiles to prevent a compile failure:

```
cp -v stage2/Makefile{,.orig}
sed '/^CFLAGS/s/-Werror //' stage2/Makefile.orig > stage2/Makefile
```

Compile the Colo package:

```
cd tools/elf2rfx
make CC=gcc
cd ../../
make CROSS_COMPILE="${CLFS_TARGET}-" binary
```

Install the package:

```
cp -v chain/colo-chain.elf /tools/boot/vmlinux
gzip -9 /tools/boot/vmlinux
```

Details on this package are located in Section 10.69.2, “Contents of Colo.”

7.12. Creating Directories

Note

The commands in the remainder of the book should be run as the `root` user. Check that `${CLFS}` is set in the `root` user's environment before proceeding.

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

```
mkdir -pv ${CLFS}/{bin,boot,dev,{etc/,}opt,home,lib/firmware,mnt}
mkdir -pv ${CLFS}/{proc,media/{floppy,cdrom},run/{,shm},sbin,svr,sys}
mkdir -pv ${CLFS}/var/{lock,log,mail,spool}
mkdir -pv ${CLFS}/var/{opt,cache,lib/{misc,locate},local}
install -dv -m 0750 ${CLFS}/root
install -dv -m 1777 ${CLFS}/{/var,}/tmp
ln -sv ../run ${CLFS}/var/run
mkdir -pv ${CLFS}/usr/{,local/}{bin,include,lib,sbin,src}
mkdir -pv ${CLFS}/usr/{,local/}share/{doc,info,locale,man}
mkdir -pv ${CLFS}/usr/{,local/}share/{misc,terminfo,zoneinfo}
mkdir -pv ${CLFS}/usr/{,local/}share/man/man{1,2,3,4,5,6,7,8}
```

These entries are needed for the RaQ2 bootloader. Only use these if you are utilizing the Colo bootloader:

```
cd ${CLFS}/boot
ln -svf . boot
```

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.

7.13. Creating Essential Symlinks

Some programs use hard-wired paths to files 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 the next chapter after the software has been installed.

```
ln -sv /tools/bin/{bash,cat,echo,grep,login,pwd,stty} ${CLFS}/bin
ln -sv /tools/bin/file ${CLFS}/usr/bin
ln -sv /tools/lib/libgcc_s.so{,.1} ${CLFS}/usr/lib
ln -sv /tools/lib/libstdc++.so{.6,} ${CLFS}/usr/lib
sed -e 's/tools/usr/' /tools/lib/libstdc++.la > ${CLFS}/usr/lib/libstdc++.la
ln -sv bash ${CLFS}/bin/sh
ln -sv /tools/sbin/init ${CLFS}/sbin
ln -sv /tools/etc/{login.{access,defs},limits} ${CLFS}/etc
```

The purpose of each link:*/bin/bash*

Many **bash** scripts specify */bin/bash*.

/bin/cat

This pathname is hard-coded into Glibc's configure script.

/bin/echo

This is to satisfy one of the tests in Glibc's test suite, which expects */bin/echo*.

/bin/grep

This to avoid a hard-coded */tools* reference in Libtool.

/bin/login

The **agetty** program expects to find **login** in */bin*.

/bin/pwd

Some **configure** scripts, particularly Glibc's, have this pathname hard-coded.

/bin/stty

This pathname is hard-coded into Expect, therefore it is needed for Binutils and GCC test suites to pass.

/usr/bin/file

Binutils' **configure** scripts specify this command location.

/usr/lib/libgcc_s.so{,.1}

Glibc needs this for the pthreads library to work.

/usr/lib/libstdc++.so{,.6}

This is needed by several tests in Glibc's test suite, as well as for C++ support in GMP.

/usr/lib/libstdc++.la

This prevents a */tools* reference that would otherwise be in */usr/lib/libstdc++.la* after GCC is installed.

/bin/sh

Many shell scripts hard-code */bin/sh*.

/sbin/init

This is where the kernel expects to find **init**.

/etc/{login.{access,defs},limits}

These are configuration files used by Shadow and are expected to be found in */etc*, for programs such as **login** and **su** to work.

Historically, Linux maintains a list of the mounted file systems in the file */etc/mtab*. Modern kernels maintain this list internally and expose it to the user via the */proc* filesystem. To satisfy utilities that expect the presence of */etc/mtab*, create the following symbolic link:

```
ln -sv /proc/self/mounts ${CLFS}/etc/mtab
```

7.14. Populating /dev

7.14.1. 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 will be created on the hard disk so that they are available before **udev** has been started, and additionally when Linux is started in single user mode (hence the restrictive permissions on console). Create these by running the following commands:

```
mknod -m 0600 ${CLFS}/dev/console c 5 1
mknod -m 0666 ${CLFS}/dev/null c 1 3
```

7.15. Creating the passwd and group Files

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 \${CLFS}/etc/passwd file by running the following command:

```
cat > ${CLFS}/etc/passwd << "EOF"
root::0:0:root:/root:/bin/bash
bin:x:1:1:/bin:/bin/false
daemon:x:2:6:/sbin:/bin/false
messagebus:x:27:27:D-Bus Message Daemon User:/dev/null:/bin/false
systemd-bus-proxy:x:71:72:systemd Bus Proxy:/:/bin/false
systemd-journal-gateway:x:73:73:systemd Journal Gateway:/:/bin/false
systemd-journal-remote:x:74:74:systemd Journal Remote:/:/bin/false
systemd-journal-upload:x:75:75:systemd Journal Upload:/:/bin/false
systemd-network:x:76:76:systemd Network Management:/:/bin/false
systemd-resolve:x:77:77:systemd Resolver:/:/bin/false
systemd-timesync:x:78:78:systemd Time Synchronization:/:/bin/false
systemd-coredump:x:79:79:systemd Core Dumper:/:/bin/false
nobody:x:65534:65533:Unprivileged User:/dev/null:/bin/false
EOF
```

The actual password for root (the “:” used here is just a placeholder and allows you to login with no password) will be set later.

Additional users you may want to add if not already included:

```
adm:x:3:16:adm:/var/adm:/bin/false
```

Was used for programs that performed administrative tasks.

```
lp:x:10:9:lp:/var/spool/lp:/bin/false
```

Used by programs for printing

```
mail:x:30:30:mail:/var/mail:/bin/false
```

Often used by email programs

```
news:x:31:31:news:/var/spool/news:/bin/false
```

Often used for network news servers

```
operator:x:50:0:operator:/root:/bin/bash
```

Often used to allow system operators to access the system

```
postmaster:x:51:30:postmaster:/var/spool/mail:/bin/false
```

Generally used as an account that receives all the information of troubles with the mail server

Create the `${CLFS}/etc/group` file by running the following command:

```
cat > ${CLFS}/etc/group << "EOF"
root:x:0:
bin:x:1:
sys:x:2:
kmem:x:3:
tty:x:5:
tape:x:4:
daemon:x:6:
floppy:x:7:
disk:x:8:
lp:x:9:
dialout:x:10:
audio:x:11:
video:x:12:
utmp:x:13:
usb:x:14:
cdrom:x:15:
adm:x:16:
mail:x:30:
messagebus:x:27:
nogroup:x:65533:
systemd-bus-proxy:x:72:
systemd-journal:x:28:
systemd-journal-gateway:x:73:
systemd-journal-remote:x:74:
systemd-journal-upload:x:75:
systemd-network:x:76:
systemd-resolve:x:77:
systemd-timesync:x:78:
systemd-coredump:x:79:
wheel:x:39:
EOF
```

Additional groups you may want to add if not already included:

```
console:x:17:
```

This group has direct access to the console

```
cdrw:x:18:
```

This group is allowed to use the CDRW drive

```
news:x:31:news
```

Used by Network News Servers

```
users:x:1000:
```

The default GID used by shadow for new users

nobody:x:65533:
This is used by NFS

The created groups are not part of any standard—they are groups decided on in part by the requirements of the Systemd configuration in the final system, and in part by common convention employed by a number of existing Linux distributions. The Linux Standard Base (LSB, available at <http://www.linuxfoundation.org/collaborate/workgroups/lsb>) 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.

7.16. 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, which must be checked, and in which order. Create a new file systems table like this:

```
cat > ${CLFS}/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
devpts         /dev/pts        devpts  gid=5,mode=620          0     0
shm            /dev/shm        tmpfs   defaults                 0     0

# End /etc/fstab
EOF
```

Replace `[xxx]`, `[yyy]`, and `[fff]` with the values appropriate for the system, for example, `sda2`, `sda5`, and `ext2`. For details on the six fields in this file, see **man 5 fstab**.

The `/dev/shm` mount point for `tmpfs` is included to allow enabling POSIX-shared memory. The kernel must have the required support built into it for this to work (more about this is in the next section). Please note that very little software currently uses POSIX-shared memory. Therefore, consider the `/dev/shm` mount point optional. For more information, see `Documentation/filesystems/tmpfs.txt` in the kernel source tree.

7.17. Setting Up the Environment

The new instance of the shell that will start when the system is booted is a *login* shell, which will read the `.bash_profile` file. Create `.bash_profile` now:

```
cat > ${CLFS}/root/.bash_profile << "EOF"
set +h
PS1='\u:\w\$ '
LC_ALL=POSIX
PATH=/bin:/usr/bin:/sbin:/usr/sbin:/tools/bin:/tools/sbin
export LC_ALL PATH PS1
EOF
```

The `LC_ALL` variable controls the localization of certain programs, making their messages follow the conventions of a specified country. Setting `LC_ALL` to “POSIX” or “C” (the two are equivalent) ensures that everything will work as expected on your temporary system.

By putting `/tools/bin` and `/tools/sbin` at the end of the standard `PATH`, all the programs installed in Constructing a Temporary System are only picked up by the shell if they have not yet been built on the target system. This configuration forces use of the final system binaries as they are built over the temp-system, minimising the chance of final system programs being built against the temp-system.

7.18. Changing Ownership

Currently, the `/tools` and `/cross-tools` directories are owned by the user `clfs`, a user that exists only on the host system. Although `/tools` and `/cross-tools` can be deleted once the CLFS system has been finished, they can be retained to build additional CLFS systems. If the `/tools` and `/cross-tools` directories are 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 these directories and all the files therein, thus exposing those files to possible malicious manipulation.

One possible fix for this issue might be to add the `clfs` user to the new CLFS system later when creating the `/etc/passwd` file, taking care to assign it the same user and group IDs as on the host system. Alternatively, assign the contents of the `/tools` and `/cross-tools` directories to user `root` by running the following commands:

```
chown -Rv 0:0 ${CLFS}/tools
chown -Rv 0:0 ${CLFS}/cross-tools
```

7.19. How to View the Book

Most likely, you have been using a web browser or PDF viewer to read the CLFS book so far. However, the temporary system in `/tools` does not have any of these, so you will need to find a way to continue following the book after booting into the temporary build environment. Possible solutions include:

- Simply have the book open on another computer, or even read a printed copy, though one downside to this is that you cannot copy-and-paste commands.
- Convert the CLFS book into plain text, thus allowing it to be viewed with **more** or **view**, by using a command such as the following:

```
lynx -dump /path/to/clfs/book.html > ${CLFS}/root/CLFS-book.txt
```

- Cross-compile and install additional programs before booting, such as Lynx or Links to view the book, or Dropbear to allow remote login. See the CLFS Hints website at <http://hints.clfs.org/index.php/> for more suggestions by other users.

7.20. Making the Temporary System Bootable

Note

This bootloader is for the MIPS based cobalt servers RaQ, RaQ2, Qube, or the Qube2.

Boot loading can be a complex area, so a few cautionary words are in order. Be familiar with the current boot loader and any other operating systems present on the hard drive(s) that need to be bootable. Make sure that an emergency boot disk is ready to “rescue” the computer if the computer becomes unusable (un-bootable).

Earlier, we compiled and installed the Cobalt boot loader software in preparation for this step. Now we will configure our system to boot using Colo. Here is a simple `default.colo` to use.

```
cat > ${CLFS}/boot/default.colo << "EOF"
#:CoLo:~
#
# load linux
#
lcd 'Booting 4.9.21....'
load vmlinux-4.9.21.gz
execute root=/dev/hda2 console=ttyS0,115200 idel=noprobe
EOF
```

7.21. What to do next

Now you're at the point to get your `${CLFS}` directory copied over to your target machine. The easiest method would be to tar it up and copy the file.

```
tar -jcvf ${CLFS}.tar.bz2 ${CLFS}
```

Chapter 8. If You Are Going to Chroot

8.1. Introduction

This chapter shows how to prepare a **chroot** jail to build the final system packages into.

8.2. Mounting Virtual Kernel File Systems

Note

The commands in the remainder of the book should be run as the `root` user. Check that `${CLFS}` is set in the `root` user's environment before proceeding.

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

Two device nodes, `/dev/console` and `/dev/null`, are required to be present on the file system. These are needed by the kernel even before starting Udev early in the boot process, so we create them here:

```
mknod -m 600 ${CLFS}/dev/console c 5 1  
mknod -m 666 ${CLFS}/dev/null c 1 3
```

Once the system is complete and booting, the rest of our device nodes will be created by the kernel's `devtmpfs` file system. For now though, we will just use the “bind” option in the `mount` command to make our host system's `/dev` structure appear in the new CLFS file system:

```
mount -v -o bind /dev ${CLFS}/dev
```

Now mount the remaining file systems:

```
mount -vt devpts -o gid=5,mode=620 devpts ${CLFS}/dev/pts  
mount -vt proc proc ${CLFS}/proc  
mount -vt tmpfs tmpfs ${CLFS}/run  
mount -vt sysfs sysfs ${CLFS}/sys
```

On some host systems, `/dev/shm` is a symbolic link to `/run/shm`. If it is, create a directory in `/run`:

```
[ -h ${CLFS}/dev/shm ] && mkdir -pv ${CLFS}/${readlink ${CLFS}/dev/shm}
```

Remember that if for any reason you stop working on the CLFS system and start again later, it is important to check that these file systems are mounted again before entering the `chroot` environment.

8.3. Before Entering the Chroot Environment

8.3.1. Determining if steps need to be taken

Before we can enter the chroot we have to make sure that the system is in the proper state. From this point on the `${CLFS_TARGET}` environment variable will no longer exist, so it will have no bearing on the rest of the book - most packages will rely on **config.guess** provided by Section 10.38, “Automake-1.15”. Packages that do not use autotools either do not care about the target triplet, or have their own means of determining its value.

In both cases, the information about the host cpu used to determine the target triplet is gathered from the same place, **uname -m**. Executing this command outside of the chroot as well as inside the chroot will have the exact same output.

If you're unsure if your host and target have the same target triplet, you can use this test to determine what the host's target triplet is and if you need to take any steps to ensure that you don't build for the wrong architecture. Extract the Section 10.38, “Automake-1.15” tarball and **cd** into the created directory. Then execute the following to see what the detected target triplet is by **config.guess**:

```
lib/config.guess
```

If the output of that command does not equal what is in `${CLFS_TARGET}` then you need to read on. If it does then you can safely continue onto Section 8.4, “Entering the Chroot Environment”.

8.3.2. Using Setarch

If your host has a tool called **setarch**, this may solve your problems, at least if you're building for i686. On an architecture such as x86_64, using **setarch linux32 uname -m** will only ever output i686. It is not possible to get an output of i486 or i586.

To test if setarch does everything you need it to, execute the following command from inside the Section 10.38, “Automake-1.15” directory:

```
setarch linux32 lib/config.guess
```

If the output of the command above equals what is in `${CLFS_TARGET}` then you have a viable solution. You can wrap the chroot command on the next page with **setarch linux32**. It will look like the following:

```
setarch linux32 chroot "${CLFS}" /tools/bin/env -i \
  HOME=/root TERM="${TERM}" PS1='\u:\w\$ ' \
  PATH=/bin:/usr/bin:/sbin:/usr/sbin:/tools/bin \
  /tools/bin/bash --login +h
```

If setarch works for you then you can safely continue onto Section 8.4, “Entering the Chroot Environment”. If not, there is one more option covered in this book.

8.3.3. Using a Uname Hack

The Uname Hack is a kernel module that modifies the output of **uname -m** by directly changing the value of the detected machine type. The kernel module will save the original value and restore it when the module is unloaded.

- **Uname Hack (20080713) - 4 KB:**

Download: http://clfs.org/files/extras/uname_hack-20080713.tar.bz2

MD5 sum: dd7694f28ccc6e6bfb326b1790adb5e9

Extract the tarball and **cd** into the created directory. To build the Uname Hack you must have the kernel sources for your currently running kernel available. Build the Uname Hack with the following or similar command:

```
make uname_hack_fake_machine=mips
```

The meaning of the make and install options:

```
uname_hack_fake_machine=mips
```

This parameter sets the value that the uts machine type will be changed to.

In the top level directory of the Uname Hack package you should see a file named `uname_hack.ko`. As soon as that module is loaded into the running kernel the output of **uname -m** will be affected immediately system-wide. Load the kernel module with the following command:

```
insmod uname_hack.ko
```

To test if the Uname Hack is working properly, execute the following command from inside the Section 10.38, “Automake-1.15” directory:

```
lib/config.guess
```

The output of the above command should be the same as the `${CLFS_TARGET}` environment variable. If this is not the case, you can try and get help on the CLFS Support Mailing List or the IRC Channel. See Section 1.7, “Help” for more information.

8.4. Entering the Chroot Environment

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

```
chroot "${CLFS}" /tools/bin/env -i \  
    HOME=/root TERM="${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 `CLFS` variable anymore, because all work will be restricted to the CLFS file system. This is because the Bash shell is told that `${CLFS}` 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**.

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), remember to first mount the `proc` and `devpts` file systems (discussed in the previous section) and enter chroot again before continuing with the installations.

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

8.5. Changing Ownership

Currently, the `/tools` and `/cross-tools` directories are owned by the user `clfs`, a user that exists only on the host system. Although `/tools` and `/cross-tools` can be deleted once the CLFS system has been finished, they can be retained to build additional CLFS systems. If the `/tools` and `/cross-tools` directories are 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 these directories and all the files therein, thus exposing those files to possible malicious manipulation.

One possible fix for this issue might be to add the `clfs` user to the new CLFS system later when creating the `/etc/passwd` file, taking care to assign it the same user and group IDs as on the host system. Alternatively, assign the contents of the `/tools` and `/cross-tools` directories to user `root` by running the following commands:

```
chown -Rv 0:0 /tools
chown -Rv 0:0 /cross-tools
```

The commands use `0:0` instead of `root:root`, because **chown** is unable to resolve the name “root” until the `passwd` file has been created.

8.6. Creating Directories

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

```
mkdir -pv /{bin,boot,dev,{etc/,}opt,home,lib,mnt}
mkdir -pv /{proc,media/{floppy,cdrom},run/shm,sbin,svr,sys}
mkdir -pv /var/{lock,log,mail,spool}
mkdir -pv /var/{opt,cache,lib/{misc,locate},local}
install -dv -m 0750 /root
install -dv -m 1777 {/var,}/tmp
ln -sv ../run /var/run
mkdir -pv /usr/{,local/}{bin,include,lib,sbin,src}
mkdir -pv /usr/{,local/}share/{doc,info,locale,man}
mkdir -pv /usr/{,local/}share/{misc,terminfo,zoneinfo}
mkdir -pv /usr/{,local/}share/man/man{1..8}
```

These entries are needed for the RaQ2 bootloader. Only use these if you are utilizing the Colo bootloader:

```
cd /boot
ln -svf . boot
```

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.

8.6.1. FHS Compliance Note

The directory tree is based on the Filesystem Hierarchy Standard (FHS) (available at <https://wiki.linuxfoundation.org/en/FHS>). In addition to the tree created above, this standard 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.

8.7. Creating Essential Symlinks

Some programs use hard-wired paths to files 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 the next chapter after the software has been installed.

```
ln -sv /tools/bin/{bash,cat,echo,grep,pwd,stty} /bin
ln -sv /tools/bin/file /usr/bin
ln -sv /tools/lib/libgcc_s.so{,.1} /usr/lib
ln -sv /tools/lib/libstdc++.so{.6,} /usr/lib
sed -e 's/tools/usr/' /tools/lib/libstdc++.la > /usr/lib/libstdc++.la
ln -sv bash /bin/sh
```

The purpose of each link:

`/bin/bash`

Many **bash** scripts specify `/bin/bash`.

`/bin/cat`

This pathname is hard-coded into Glibc's configure script.

`/bin/echo`

This is to satisfy one of the tests in Glibc's test suite, which expects `/bin/echo`.

`/bin/grep`

This to avoid a hard-coded `/tools` reference in Libtool.

`/bin/pwd`

Some **configure** scripts, particularly Glibc's, have this pathname hard-coded.

`/bin/stty`

This pathname is hard-coded into Expect, therefore it is needed for Binutils and GCC test suites to pass.

`/usr/bin/file`

Binutils' **configure** scripts specify this command location.

`/usr/lib/libgcc_s.so{,.1}`

Glibc needs this for the pthreads library to work.

`/usr/lib/libstdc++{,.6}`

This is needed by several tests in Glibc's test suite, as well as for C++ support in GMP.

`/usr/lib/libstdc++.la`

This prevents a `/tools` reference that would otherwise be in `/usr/lib/libstdc++.la` after GCC is installed.

`/bin/sh`

Many shell scripts hard-code `/bin/sh`.

`/sbin/init`

This is where the kernel expects to find **init**.

Historically, Linux maintains a list of the mounted file systems in the file `/etc/mtab`. Modern kernels maintain this list internally and expose it to the user via the `/proc` filesystem. To satisfy utilities that expect the presence of `/etc/mtab`, create the following symbolic link:

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

8.8. Creating the passwd and group Files

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:/bin/false
daemon:x:2:6:/sbin:/bin/false
messagebus:x:27:27:D-Bus Message Daemon User:/dev/null:/bin/false
systemd-bus-proxy:x:71:72:systemd Bus Proxy:/:/bin/false
systemd-journal-gateway:x:73:73:systemd Journal Gateway:/:/bin/false
systemd-journal-remote:x:74:74:systemd Journal Remote:/:/bin/false
systemd-journal-upload:x:75:75:systemd Journal Upload:/:/bin/false
systemd-network:x:76:76:systemd Network Management:/:/bin/false
systemd-resolve:x:77:77:systemd Resolver:/:/bin/false
systemd-timesync:x:78:78:systemd Time Synchronization:/:/bin/false
systemd-coredump:x:79:79:systemd Core Dumper:/:/bin/false
nobody:x:65534:65533:Unprivileged User:/dev/null:/bin/false
EOF
```

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

Additional users you may want to add if not already included:

```
adm:x:3:16:adm:/var/adm:/bin/false
```

Was used for programs that performed administrative tasks.

```
lp:x:10:9:lp:/var/spool/lp:/bin/false
```

Used by programs for printing

```
mail:x:30:30:mail:/var/mail:/bin/false
```

Often used by email programs

```
news:x:31:31:news:/var/spool/news:/bin/false
```

Often used for network news servers

```
operator:x:50:0:operator:/root:/bin/bash
```

Often used to allow system operators to access the system

```
postmaster:x:51:30:postmaster:/var/spool/mail:/bin/false
```

Generally used as an account that receives all the information of troubles with the mail server

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

```
cat > /etc/group << "EOF"
root:x:0:
bin:x:1:
sys:x:2:
kmem:x:3:
tty:x:5:
tape:x:4:
daemon:x:6:
floppy:x:7:
disk:x:8:
lp:x:9:
dialout:x:10:
audio:x:11:
video:x:12:
utmp:x:13:
usb:x:14:
cdrom:x:15:
adm:x:16:
mail:x:30:
messagebus:x:27:
nogroup:x:65533:
systemd-bus-proxy:x:72:
systemd-journal:x:28:
systemd-journal-gateway:x:73:
systemd-journal-remote:x:74:
systemd-journal-upload:x:75:
systemd-network:x:76:
systemd-resolve:x:77:
systemd-timesync:x:78:
systemd-coredump:x:79:
wheel:x:39:
EOF
```

Additional groups you may want to add if not already included:

```
console:x:17:
```

This group has direct access to the console

```
cdrw:x:18:
```

This group is allowed to use the CDRW drive

```
news:x:31:news
```

Used by Network News Servers

```
users:x:1000:
```

The default GID used by shadow for new users

```
nobody:x:65533:
```

This is used by NFS

The created groups are not part of any standard—they are groups decided on in part by the requirements of the Systemd configuration in the final system, and in part by common convention employed by a number of existing Linux distributions. The Linux Standard Base (LSB, available at <http://www.linuxfoundation.org/collaborate/workgroups/lsb>) 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 Constructing Cross-Compile Tools 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 the next chapters.

Part V. Building the CLFS System

Chapter 9. Constructing Testsuite Tools

9.1. Introduction

This chapter builds the tools needed by some packages to run the tests that they have. I.e., **make check**. Tcl, Expect, and DejaGNU are needed for the GCC, Binutils, and Findutils test suites. Installing three 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.

9.2. Tcl-8.6.4

The Tcl package contains the Tool Command Language.

9.2.1. Installation of Tcl

Prepare Tcl for compilation:

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

Build the package:

```
make
```

Install the package:

```
make install
```

Tcl's private header files are needed for the next package, Expect. Install them into /tools:

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

Now make a necessary symbolic link:

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

9.2.2. Contents of Tcl

Installed programs:	tclsh (link to tclsh8.6) and tclsh8.6
Installed libraries:	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

9.3. Expect-5.45

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

9.3.1. Installation of Expect

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 testsuite-tools location.

`--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 the Tcl source directory.

Build the package:

```
make
```

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.

9.3.2. Contents of Expect

Installed program:	expect
Installed library:	libexpect-5.43.a

Short Descriptions

expect	Communicates with other interactive programs according to a script
libexpect-5.43.a	Contains functions that allow Expect to be used as a Tcl extension or to be used directly from C or C++ (without Tcl)

9.4. DejaGNU-1.6

The DejaGNU package contains a framework for testing other programs.

9.4.1. Installation of DejaGNU

Prepare DejaGNU for compilation:

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

Build and install the package:

```
make install
```

9.4.2. Contents of DejaGNU

Installed program: runtest

Short Descriptions

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

Chapter 10. Installing Basic System Software

10.1. Introduction

In this chapter, we enter the building site and start constructing the CLFS system in earnest. 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 the user (or the system) needs it. For every installed package, a summary of its contents is given, followed by concise descriptions of each program and library the package installed.

If using compiler optimizations, please review the optimization hint at <http://hints.clfs.org/index.php/Optimization>. Compiler optimizations 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 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 CLFS 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 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.

To keep track of which package installs particular files, a package manager can be used. For a general overview of different styles of package managers, please take a look at the next page.

10.2. Package Management

Package Management is an often-requested addition to the CLFS Book. A Package Manager allows tracking the installation of files making it easy to remove and upgrade packages. 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 specific package manager is recommended in CLFS or CBLFS 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 subproject* and see if one of them fits your need.

10.2.1. Upgrade Issues

A Package Manager makes it easy to upgrade to newer versions when they are released. Generally the instructions in CLFS and CBLFS 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 CLFS. 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.
- If you are upgrading a running system, be on the lookout for packages that use **cp** instead of **install** to install files. The latter command is usually safer if the executable or library is already loaded in memory.

10.2.2. 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.

10.2.2.1. 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.

10.2.2.2. 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.

10.2.2.3. 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`.

10.2.2.4. 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.

10.2.2.5. LD_PRELOAD Based

In this approach, a library is 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.

10.2.2.6. 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 `CLFS` systems is located at <http://hints.clfs.org/index.php/Fakeroot>.

10.3. About Test Suites, Again

In the final-system build, you are no longer cross-compiling so it is possible to run package test suites. 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.

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 (if you chrooted) is that the host system does not have the `devpts` file system set up correctly. This issue is discussed in greater detail at <http://trac.clfs.org/wiki/faq#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://clfs.org/testsuite-logs/git/> to verify whether or not these failures are expected. This site is valid for all tests throughout this book.

10.4. Temporary Perl-5.26.0

The Perl package contains the Practical Extraction and Report Language.

10.4.1. Installation of Perl

Note

In this section, we will add Perl to the temporary system in `/tools`. This package installation should technically be part of Constructing a Temporary System, but Perl has often had problems with cross-compiling, so we will compile and install it while in the final build environment.

Change a hardcoded path from `/usr/include` to `/tools/include`:

```
sed -i 's@/usr/include@/tools/include@g' ext/Errno/Errno_pm.PL
```

Prepare Temporary Perl for compilation:

```
./configure.gnu \
  --prefix=/tools \
  -Dcc="gcc"
```

The meaning of the configure option:

`-Dcc="gcc"`

Tells Perl to use `gcc` instead of the default `cc`.

Compile the package:

```
make
```

Although Perl comes with a test suite, it is not recommended to run it at this point, as this Perl installation is only temporary. The test suite can be run later in this chapter if desired.

Install the package:

```
make install
```

Finally, create a necessary symlink:

```
ln -sfv /tools/bin/perl /usr/bin
```

Details on this package are located in Section 10.35.2, “Contents of Perl.”

10.5. Linux-4.9.21 Headers

The Linux Kernel contains a **make** target that installs “sanitized” kernel headers.

10.5.1. Installation of Linux Headers

Note

For this step you will need to unpack the kernel tarball (`linux-4.9.tar.xz`) and **cd** into its source directory before entering the commands on this page.

Apply the latest Linux sublevel patch:

```
xzcat ../patch-4.9.21.xz | patch -Np1 -i -
```

Install the kernel header files:

```
make mrproper
make headers_check
make INSTALL_HDR_PATH=/usr headers_install
find /usr/include -name .install -or -name ..install.cmd | xargs rm -fv
```

The meaning of the make commands:

make mrproper

Ensures that the kernel source dir is clean.

make headers_check

Sanitizes the raw kernel headers so that they can be used by userspace programs.

make INSTALL_HDR_PATH=/usr headers_install

This will install the kernel headers into `/usr/include`.

find /usr/include -name .install -or -name ..install.cmd | xargs rm -fv

Removes a number of unneeded debugging files that were installed.

10.5.2. Contents of Linux Headers

Installed headers:	<code>/usr/include/{asm,asm-generic,drm,linux,misc,mtd,rdma,scsi,sound,video,xen}/*.h</code>
Installed directories:	<code>/usr/include/asm, /usr/include/asm-generic, /usr/include/drm, /usr/include/linux, /usr/include/misc, /usr/include/mtd, /usr/include/rdma, /usr/include/scsi, /usr/include/sound, /usr/include/uapi, /usr/include/video, /usr/include/xen</code>

Short Descriptions

<code>/usr/include/{asm,asm-generic,drm,linux,mtd,rdma,sound,video}/*.h</code>	The Linux API headers
--	-----------------------

10.6. Man-pages-4.09

The Man-pages package contains over 2,200 man pages.

10.6.1. Installation of Man-pages

Install Man-pages by running:

```
make install
```

10.6.2. Contents of Man-pages

Installed files: various man pages

Short Descriptions

`man` `pages` This package contains man pages that describe the following: POSIX headers (section 0p), POSIX utilities (section 1p), POSIX functions (section 3p), user commands (section 1), system calls (section 2), libc calls (section 3), device information (section 4), file formats (section 5), games (section 6), conventions and macro packages (section 7), system administration (section 8), and kernel (section 9).

10.7. Glibc-2.25

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.

10.7.1. Installation of Glibc

Note

Some packages outside of CLFS 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 a CLFS system.

At the end of the installation, the build system will run a sanity test to make sure everything installed properly. This script performs its tests by attempting to compile test programs against certain libraries. However it does not specify the path to `ld.so`, and our toolchain is still configured to use the one in `/tools`. The following set of commands will force the script to use the complete path of the new `ld.so` that was just installed:

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

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-2.25/configure \
--prefix=/usr \
--enable-kernel=3.12.0 \
--libexecdir=/usr/lib/glibc \
--enable-stack-protector=no \
--enable-obsolete-rpc
```

The meaning of the new configure option:

```
--libexecdir=/usr/lib/glibc
```

This changes the location for hard links to the **getconf** utility from their default of `/usr/libexec` to `/usr/lib/glibc`.

Compile the package:

```
make
```

Important

Due to Glibc's critical role in a properly functioning system, the CLFS developers strongly recommend running the testsuite.

Use the following commands to run the test suite and output any test failures:

```
make check
```

The Glibc test suite is highly dependent on certain functions of the host system, in particular the kernel. The *posix/annexc* and *conform/run-conformtest* tests normally fail and you should see `Error 1 (ignored)` in the output. Apart from this, the Glibc test suite is always expected to pass. However, in certain circumstances, some failures are unavoidable. If a test fails because of a missing program (or missing symbolic link), or a segfault, you will see an error code greater than 127 and the details will be in the log. More commonly, tests will fail with `Error 2` - for these, the contents of the corresponding `.out` file, e.g. *posix/annexc.out* may be informative. Here 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. Certain optimization settings are known to be a factor here.
- If you have mounted the CLFS partition with the *noatime* option, the *atime* test will fail. As mentioned in Section 2.5, “Mounting the New Partition”, do not use the *noatime* option while building CLFS.
- When running on older and slower hardware, 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`).
- *posix/tst-getaddrinfo4* will always fail due to not having a network connection when the test is run.

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, and remove unneeded files from `/usr/include/rpcsvc`:

```
make install &&
rm -v /usr/include/rpcsvc/*.x
```

Install the configuration file and runtime directory for **nscd**:

```
cp -v ../glibc-2.25/nscd/nscd.conf /etc/nscd.conf
mkdir -pv /var/cache/nscd
```

Install the systemd support files for **nscd**:

```
install -v -Dm644 ../glibc-2.25/nscd/nscd.tmpfiles /usr/lib/tmpfiles.d/nscd.conf
install -v -Dm644 ../glibc-2.25/nscd/nscd.service /lib/systemd/system/nscd.service
```

10.7.2. Internationalization

The locales that can make the system respond in a different language were not installed by the above command. Install them with:

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

To save time, an alternative to running the previous command (which generates and installs every locale listed in the `glibc-2.25/localedata/SUPPORTED` file) is to install only those locales that are wanted and needed. This can be achieved by using the **localedef** command. Information on this command is located in the `INSTALL` file in the Glibc source. However, there are a number of locales that are essential in order for the tests of future packages to pass, in particular, the *libstdc++* tests from GCC. The following instructions, instead of the *install-locales* target used above, will install the minimum set of locales necessary for the tests to run successfully:

```
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 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 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 it_IT -f ISO-8859-1 it_IT
localedef -i ja_JP -f EUC-JP ja_JP
```

Some locales installed by the **make localedata/install-locales** command above are not properly supported by some applications that are in CLFS and CBLFS. Because of the various problems that arise due to application programmers making assumptions that break in such locales, CLFS should not be used in locales that utilize multibyte character sets (including UTF-8) or right-to-left writing order. Numerous unofficial and unstable patches are required to fix these problems, and it has been decided by the CLFS developers not to support such complex locales at this time. This applies to the `ja_JP` and `fa_IR` locales as well—they have been installed only for GCC and Gettext tests to pass, and the **watch** program (part of the *Procps-ng* package) does not work properly in them. Various attempts to circumvent these restrictions are documented in internationalization-related hints.

10.7.3. 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 ../tzdata2017b.tar.gz

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

for tz in etcetera southamerica northamerica europe africa antarctica \
    asia australasia backward pacificnew systemv; do
    zic -L /dev/null -d $ZONEINFO -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 zone1970.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.

To determine the local time zone, run the following script:

```
tzselect
```

After answering a few questions about the location, the script will output the name of the time zone (e.g., *EST5EDT* or *Canada/Eastern*). Then create the `/etc/localtime` file by running:

```
cp -v /usr/share/zoneinfo/[xxx] \  
    /etc/localtime
```

Replace `[xxx]` with the name of the time zone that **tzselect** provided (e.g., *Canada/Eastern*).

10.7.4. Configuring The Dynamic Loader

By default, the dynamic loader (`/lib/ld.so.1`) searches through `/lib` and `/usr/lib` for dynamic libraries that are needed by programs as they are run. However, if there are libraries in directories other than `/lib` and `/usr/lib`, these need to be added to the `/etc/ld.so.conf` file in order for the dynamic loader to find them. Two directories that are commonly known to contain additional libraries are `/usr/local/lib` and `/opt/lib`, so add those directories to the dynamic loader's search path.

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

```
cat > /etc/ld.so.conf << "EOF"  
# Begin /etc/ld.so.conf  
  
/usr/local/lib  
/opt/lib  
  
# End /etc/ld.so.conf  
EOF
```

10.7.5. Contents of Glibc

Installed programs:	catchsegv, gencat, getconf, getent, iconv, iconvconfig, ldconfig, ldd, lddlibc4, locale, localedef, makedb, mtrace, nsd, pcprofiledump, pldd, rpcgen, sln, sotruss, sprof, tzselect, xtrace, zdump, zic
Installed libraries:	ld.so, libBrokenLocale.[a,so], libSegFault.so, libanl.[a,so], libc.[a,so], libc_nonshared.a, libcidn.[a,so], libcrypt.[a,so], libdl.[a,so], libg.a, libieee.a, libm.[a,so], libmcheck.a, libmemusage.so, libnsl.a, 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, 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/scsi, /usr/include/sys, /usr/lib/audit, /usr/lib/gconv, /usr/lib/glibc, /usr/lib/locale, /usr/share/i18n, /usr/share/zoneinfo, /var/cache/ldconfig, /var/cache/nsd

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	Tells the compiler to enable or disable the use of POSIX locales for built-in operations
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
nsd	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
rpcgen	Generates C code to implement the Remote Procedure Call (RPC) protocol
sln	A statically linked program that creates symbolic links
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 by programs, such as Mozilla, to solve broken locales
libSegFault	The segmentation fault signal handler
libanl	An asynchronous name lookup library
libc	The main C library
libcidn	Used internally by Glibc for handling internationalized domain names in the <code>getaddrinfo()</code> function
libcrypt	The cryptography library
libdl	The dynamic linking interface library
libg	A runtime library for g++
libieee	The Institute of Electrical and Electronic Engineers (IEEE) floating point library
libm	The mathematical library
libmcheck	Contains code run at boot
libmemusage	Used by memusage (included in Glibc, but not built in a base CLFS system as it has additional dependencies) 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

10.8. Adjusting the Toolchain

Now we adjust GCC's specs so that they point to the new dynamic linker. A **perl** command accomplishes this:

```
gcc -dumpspecs | \
perl -p -e 's@/tools/lib/ld@/lib/ld@g;' \
-e 's@\*startfile_prefix_spec:\n@$_/usr/lib/ @g;' > \
$(dirname $(gcc --print-libgcc-file-name))/specs
```

The **perl** command above makes 2 modifications to GCC's specs: it removes “/tools” from the pathname to the dynamic linker, and adds “/usr/lib/” to the `startfile_prefix_spec`. It is a good idea to visually inspect the `specs` file, and compare with the output of **gcc -dumpspecs**, to verify that the intended changes were actually made.

Caution

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

```
echo 'int main(){}' > dummy.c
gcc dummy.c
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:

```
[Requesting program interpreter: /lib/ld.so.1]
```

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

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 amendment above. 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
```

10.9. M4-1.4.18

The M4 package contains a macro processor.

10.9.1. Installation of M4

Prepare M4 for compilation:

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

Compile the package:

```
make
```

To test the results, issue:

```
make check
```

Install the package:

```
make install
```

10.9.2. 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.

10.10. GMP-6.1.2

GMP is a library for arithmetic on arbitrary precision integers, rational numbers, and floating-point numbers.

10.10.1. Installation of GMP

Note

If you are compiling this package on a different CPU than you plan to run the CLFS system on, you must replace GMP's `config.guess` and `config.sub` wrappers with the originals. This will prevent GMP from optimizing for the wrong CPU. You can make this change with the following command:

```
mv -v config{fsf,}.guess
mv -v config{fsf,}.sub
```

Prepare GMP for compilation:

```
CC="gcc -isystem /usr/include" \
CXX="g++ -isystem /usr/include" \
LDFLAGS="-Wl,-rpath-link,/usr/lib:/lib" \
./configure \
  --prefix=/usr \
  --enable-cxx \
  --docdir=/usr/share/doc/gmp-6.1.2
```

Compile the package:

```
make
```

Build the HTML documentation:

```
make html
```

Test the results:

```
make check
```

Install the package:

```
make install
```

Install the documentation:

```
make install-html
```

10.10.2. Contents of GMP

Installed libraries:	<code>libgmp.[a,so]</code> , <code>libgmpxx.[a,so]</code>
Installed directory:	<code>/usr/share/doc/gmp-6.1.2</code>

Short Descriptions

`libgmp` Contains the definitions for GNU multiple precision functions.

`libgmpxx` Contains a C++ class wrapper for GMP types.

10.11. MPFR-3.1.5

The MPFR library is a C library for multiple-precision floating-point computations with correct rounding.

10.11.1. Installation of MPFR

Apply a patch with upstream fixes:

```
patch -Np1 -i ../mpfr-3.1.5-fixes-1.patch
```

Prepare MPFR for compilation:

```
CC="gcc -isystem /usr/include" \
LDFLAGS="-Wl,-rpath-link,/usr/lib:/lib" \
./configure \
  --prefix=/usr \
  --with-gmp=/usr \
  --docdir=/usr/share/doc/mpfr-3.1.5
```

Compile the package:

```
make
make html
```

Test the results:

```
make check
```

Install the package:

```
make install
make install-html
```

10.11.2. Contents of MPFR

Installed libraries:	libmpfr.[a,so]
Installed directory:	/usr/share/doc/mpfr-3.1.5

Short Descriptions

`libmpfr` The Multiple Precision Floating-Point Reliable Library.

10.12. MPC-1.0.3

MPC is a C library for the arithmetic of complex numbers with arbitrarily high precision and correct rounding of the result.

10.12.1. Installation of MPC

Prepare MPC for compilation:

```
CC="gcc -isystem /usr/include" \
LDFLAGS="-Wl,-rpath-link,/usr/lib:/lib" \
./configure \
    --prefix=/usr \
    --docdir=/usr/share/doc/mpc-1.0.3
```

Compile the package:

```
make
```

Build the HTML documentation:

```
make html
```

Test the results:

```
make check
```

Install the package:

```
make install
```

Install the HTML documentation:

```
make install-html
```

10.12.2. Contents of MPC

Installed libraries:	libmpc.[a,so]
Installed directory:	/usr/share/doc/mpc-1.0.3

Short Descriptions

`libmpc` The Multiple Precision Complex Library.

10.13. ISL-0.17.1

ISL is a library for manipulating sets and relations of integer points bounded by linear constraints.

10.13.1. Installation of ISL

Prepare ISL for compilation:

```
CC="gcc -isystem /usr/include" \
LDFLAGS="-Wl,-rpath-link,/usr/lib:/lib" \
./configure \
    --prefix=/usr
```

Compile the package:

```
make
```

Test the results:

```
make check
```

Install the package:

```
make install
```

Finally, move a misplaced file:

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

10.13.2. Contents of ISL

Installed libraries:	libisl.[a,so]
Installed directory:	/usr/include/isl

Short Descriptions

`libisl` The Integer Set Library.

10.14. Zlib-1.2.11

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

10.14.1. Installation of Zlib

Prepare Zlib for compilation:

```
CC="gcc -isystem /usr/include" \
CXX="g++ -isystem /usr/include" \
LDFLAGS="-Wl,-rpath-link,/usr/lib:/lib" \
./configure \
    --prefix=/usr
```

Compile the package:

```
make
```

To test the results, issue:

```
make check
```

Install the package:

```
make install
```

The previous command installed two `.so` files into `/usr/lib`. We will move them into `/lib` and then recreate a link in `/usr/lib`:

```
mv -v /usr/lib/libz.so.* /lib
ln -sfv ../../lib/${readlink /usr/lib/libz.so} /usr/lib/libz.so
```

Install the documentation:

```
mkdir -pv /usr/share/doc/zlib-1.2.11
cp -rv doc/* examples /usr/share/doc/zlib-1.2.11
```

10.14.2. Contents of Zlib

Installed libraries:	libz.[a,so]
Installed directory:	/usr/share/doc/zlib-1.2.11

Short Descriptions

`libz` Contains compression and decompression functions used by some programs

10.15. Flex-2.6.4

The Flex package contains a utility for generating programs that recognize patterns in text.

10.15.1. Installation of Flex

Prepare Flex for compilation:

```
./configure \
  --prefix=/usr \
  --docdir=/usr/share/doc/flex-2.6.4
```

Compile the package:

```
make
```

To test the results, issue:

```
make check
```

The test suite will report 3 failures for tests that use **bison**, which is not installed yet. For full test coverage, you can run Flex's test suite again after Bison is installed.

Install the package:

```
make install
```

A few programs do not know about **flex** yet and try to run its predecessor, **lex**. To support those programs, create a symbolic link named `lex` that runs `flex` in **lex** emulation mode:

```
ln -sv flex /usr/bin/lex
```

10.15.2. Contents of Flex

Installed programs:	flex, flex++ (link to flex), lex
Installed libraries:	libfl.[a,so], libfl_pic.[a,so]
Installed directory:	/usr/share/doc/flex-2.6.4

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++	Link to flex which makes it generate C++ scanner classes
lex	A script that runs flex in lex emulation mode
libfl	The flex library
libfl_pic	The flex library

10.16. Bison-3.0.4

The Bison package contains a parser generator.

10.16.1. Installation of Bison

Prepare Bison for compilation:

```
./configure \
    --prefix=/usr \
    --docdir=/usr/share/doc/bison-3.0.4
```

Compile the package:

```
make
```

To test the results, issue:

```
make check
```

Install the package:

```
make install
```

10.16.2. Contents of Bison

Installed programs:	bison, yacc
Installed library:	liby.a
Installed directories:	/usr/share/bison, /usr/share/doc/bison-3.0.4

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 <code>-y</code> option
liby.a	The Yacc library containing implementations of Yacc-compatible <i>yyerror</i> and <i>main</i> functions; this library is normally not very useful, but POSIX requires it

10.17. Binutils-2.28

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

10.17.1. Installation of Binutils

Verify that the PTYs are working properly inside the build environment. Check that everything is set up correctly by performing a simple test:

```
expect -c "spawn ls"
```

This command should give the following output:

```
spawn ls
```

If, instead, it gives a message saying to create more ptys, 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 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:

```
CC="gcc -isystem /usr/include" \
LD_FLAGS="-Wl,-rpath-link,/usr/lib:/lib" \
../binutils-2.28/configure \
  --prefix=/usr \
  --enable-shared \
  --enable-gold=yes \
  --enable-plugins \
  --with-system-zlib \
  --enable-threads
```

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)`. Because this is a custom system, this target-specific directory in `/usr` is not required.

Important

Due to Binutils' critical role in a properly functioning system, the CLFS developers strongly recommend running the testsuite.

Test the results:

```
make check
```

Install the package:

```
make tooldir=/usr install
```

10.17.2. Contents of Binutils

Installed programs: addr2line, ar, as, c++filt, elfedit, gprof, ld, ld.bfd, ld.gold, nm, objcopy, objdump, ranlib, readelf, size, strings, strip

Installed libraries: libbfd.[a,so], 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
ld.gold	A linker designed to be faster than ld , especially for large C++ applications.
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
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**.

10.18. GCC-7.1.0

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

10.18.1. Installation of GCC

Apply a **sed** substitution that will suppress the execution of the **fixincludes** script:

```
sed -i 's@\.\/fixinc\.sh@-c true@' gcc/Makefile.in
```

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

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

Prepare GCC for compilation:

```
SED=sed CC="gcc -isystem /usr/include" \
CXX="g++ -isystem /usr/include" \
LDFLAGS="-Wl,-rpath-link,/usr/lib:/lib" \
../gcc-7.1.0/configure \
  --prefix=/usr \
  --libexecdir=/usr/lib \
  --enable-languages=c,c++ \
  --disable-multilib \
  --with-system-zlib \
  --enable-install-libiberty \
  --disable-bootstrap
```

The meaning of the new configure options:

SED=sed

This prevents a hard-coded path to `/tools/bin/sed` in the **fixincl** program.

--disable-bootstrap

For a native build, GCC defaults to performing a 3-stage "bootstrap" of the compiler. This means that GCC is compiled a total of 3 times - it is compiled once, the first stage compiler is used to build itself again, and the second stage compiler builds itself once more. The second and third passes are then compared, verifying that GCC is able to reproduce itself successfully. However, there is no need for this with the CLFS build process so we disable it here.

Compile the package:

```
make
```

Important

Due to GCC's critical role in a properly functioning system, the CLFS developers strongly recommend running the testsuite.

Increase the stack size prior to running the tests:

```
ulimit -s 32768
```

Test the results, but do not stop at errors:

```
make -k check
```

The `-k` flag is used to make the test suite run through to completion and not stop at the first failure. The GCC test suite is very comprehensive and is almost guaranteed to generate a few failures. To receive a summary of the test suite results, run:

```
../gcc-7.1.0/contrib/test_summary
```

For only the summaries, pipe the output through **grep -A7 Summ.**

A few unexpected failures cannot always be avoided. The GCC developers are usually aware of these issues, but have not resolved them yet.

Install the package:

```
make install
```

Create a link to satisfy FHS requirements:

```
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
```

Finally, move a misplaced file:

```
mv -v /usr/lib/libstdc++*gdb.py /usr/share/gdb/auto-load/usr/lib
```

10.18.2. Contents of GCC

Installed programs:	c++, cc (link to gcc), cpp, g++, gcc, gcov, gcov-tool
Installed libraries:	libasan.[a,so], libatomic.[a,so], libcc1.so, libcilkrts.[a,so], libgcc.a, libgcc_eh.a, libgcc_s.so, libgcov.a, libgomp.[a,so], libiberty.a, libitm.[a,so], liblsan.[a,so], liblto_plugin.so, libquadmath.[a,so], libssp.[a,so], libssp_nonshared.a, libstdc++.a, libsupc++.a, libtsan.[a,so], libubsan.[a,so], libvtv.[a,so]
Installed directories:	/usr/include/[c++,libiberty], /usr/lib/gcc, /usr/share/gcc-7.1.0

Short Descriptions

cc	The C compiler
cpp	The C preprocessor; it is used by the compiler to expand the <code>#include</code> , <code>#define</code> , and similar statements in the source files
c++	The C++ compiler
g++	The C++ compiler
gcc	The C compiler
gcov	A coverage testing tool; it is used to analyze programs to determine where optimizations will have the most effect
gcov-tool	An offline tool to handle gcda counts

<code>libasan</code>	The Address Sanitizer runtime library
<code>libatomic</code>	A GCC support runtime library for atomic operations not supported by hardware
<code>libcc1</code>	Translates API into RPC calls
<code>libcilkrts</code>	Intel® Cilk™ Plus runtime library
<code>libgcc</code>	Contains run-time support for gcc
<code>libgcov</code>	Library that is linked into a program when gcc is instructed to enable profiling
<code>libgomp</code>	GNU implementation of the OpenMP API for multi-platform shared-memory parallel programming in C/C++ and Fortran
<code>libiberty</code>	Contains routines used by various GNU programs, including getopt , obstack , strerror , strtol , and strtoul
<code>libitm</code>	The GNU Transactional Memory Library, which provides transaction support for accesses to a process's memory
<code>liblsan</code>	The Leak Sanitizer runtime library
<code>liblto_plugin</code>	Runtime library for GCC's link-time optimization plugin
<code>libquadmath</code>	The GCC Quad-Precision Math Library API
<code>libssp</code>	Contains routines supporting GCC's stack-smashing protection functionality
<code>libstdc++</code>	The standard C++ library
<code>libsupc++</code>	Provides supporting routines for the C++ programming language
<code>libtsan</code>	The Thread Sanitizer runtime library
<code>libubsan</code>	The Undefined Behavior Sanitizer runtime library
<code>libvtv</code>	The Virtual Table Verification runtime library

10.19. Attr-2.4.47

Attr is a library for getting and setting POSIX.1e (formerly POSIX 6) draft 15 capabilities.

10.19.1. Installation of Attr

Apply a sed which prevents man-pages which were installed by the Section 10.6, “Man-pages-4.09” package:

```
sed -i -e "/SUBDIRS/s|man[25]||g" man/Makefile
```

Apply a sed to install the documentation with a versioned directory:

```
sed -i -e 's|/@pkg_name@|&-@pkg_version@|' include/builddefs.in
```

Prepare Attr for compilation:

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

Compile the package:

```
make
```

The tests need to run with a filesystem which supports extended attributes. Test the results:

```
make -j1 tests root-tests
```

Install the package:

```
make install install-dev install-lib
```

Move the shared library to `/lib` and recreate the symlink in `/usr/lib` :

```
mv -v /usr/lib/libattr.so.* /lib
ln -sfv ../../lib/$(readlink /usr/lib/libattr.so) /usr/lib/libattr.so
```

Set the proper permissions on the shared library:

```
chmod 755 -v /lib/libattr.so.1.1.0
```

10.19.2. Contents of Attr

Installed programs:	attr, getfattr, setfattr
Installed libraries:	libattr.[a,so]
Installed directories:	/usr/include/attr, /usr/share/doc/attr-2.4.47

Short Descriptions

attr	Manage extended attributes on filesystem objects
getfattr	Get extended attributes of filesystem objects
setfattr	Set extended attributes of filesystem objects
libattr	Library to manage extended attributes on filesystem objects

10.20. Acl-2.2.52

Acl is a library for getting and setting POSIX Access Control Lists.

10.20.1. Installation of Acl

Apply a sed for a test:

```
sed -i -e "/TABS-1;/a if (x > (TABS-1)) x = (TABS-1);" \
libacl/__acl_to_any_text.c
```

Apply a sed to install the documentation with a versioned directory:

```
sed -i -e 's|/@pkg_name@|&-@pkg_version@|' include/builddefs.in
```

Apply a sed to fix a few tests:

```
sed -i "s:| sed.*::g" test/{sbits-restore,cp,misc}.test
```

Prepare Acl for compilation:

```
./configure \
  --prefix=/usr \
  --libexecdir=/usr/lib
```

Compile the package:

```
make
```

The Acl tests need a filesystem which supports access controls after Coreutils has been built with the Acl libraries. Return to this section after Coreutils has been installed. Test the results:

```
make tests
```

Install the package:

```
make install install-dev install-lib
```

Move the shared library to /lib and recreate the symlink in /usr/lib:

```
mv -v /usr/lib/libacl.so.* /lib
ln -sfv ../../lib/libacl.so.1 /usr/lib/libacl.so
```

Set the proper permissions on the shared library:

```
chmod 755 -v /lib/libacl.so.1.1.0
```

10.20.2. Contents of Acl

Installed programs:	chacl, getfacl, setfacl
Installed libraries:	libattr.[a,so]
Installed directories:	/usr/include/acl, /usr/share/doc/acl-2.2.52

Short Descriptions

chacl Changes the access control list of a file or directory

getfacl	Get file access control lists
setfacl	Set file access control lists
libacl	Library to manage access control lists

10.21. Libcap-2.25

Libcap is a library for getting and setting POSIX.1e (formerly POSIX 6) draft 15 capabilities.

10.21.1. Installation of Libcap

Compile the package:

```
make
```

This package does not come with a test suite.

Install the package:

```
make RAISE_SETFCAP=no install
chmod -v 755 /lib/libcap.so.2.25
ln -sfv ../../lib/$(readlink /lib/libcap.so) /usr/lib/libcap.so
rm -v /lib/libcap.so
mv -v /lib/libcap.a /usr/lib
```

The meaning of the make option:

RAISE_SETFCAP=no

This prevents **setcap** from being run on itself, which will fail if the kernel or file system does not support extended capabilities.

10.21.2. Contents of Libcap

Installed programs:	capsh, getcap, getpcaps, setcap
Installed libraries:	libcap.[a,so]

Short Descriptions

capsh	Capability support and use can be explored and constrained with this tool
getcap	Examines file capabilities
getpcaps	Displays the capabilities on the queried process(es)
setcap	Sets file capabilities
libcap	Library for setting and clearing POSIX.1e capabilities

10.22. Sed-4.4

The Sed package contains a stream editor.

10.22.1. Installation of Sed

Prepare Sed for compilation:

```
./configure \  
  --prefix=/usr \  
  --bindir=/bin \  
  --docdir=/usr/share/doc/sed-4.4
```

Compile the package:

```
make
```

Build the HTML documentation:

```
make html
```

To test the results, issue:

```
make check
```

Install the package:

```
make install
```

Install the HTML documentation:

```
make install-html-am
```

10.22.2. Contents of Sed

Installed program:	sed
Installed directory:	/usr/share/doc/sed-4.4

Short Descriptions

sed Filters and transforms text files in a single pass

10.23. Pkg-config-lite-0.28-1

Pkg-config-lite is a tool to help you insert the correct compiler options on the command line when compiling applications and libraries.

10.23.1. Installation of Pkg-config-lite

Prepare Pkg-config-lite for compilation:

```
./configure \
  --prefix=/usr \
  --docdir=/usr/share/doc/pkg-config-0.28-1
```

Compile the package:

```
make
```

To test the results, issue:

```
make check
```

Install the package:

```
make install
```

10.23.2. Contents of Pkg-config-lite

Installed programs:	pkg-config
Installed directory:	/usr/share/doc/pkg-config-0.28-1

Short Descriptions

pkg-config	The pkg-config program is used to retrieve information about installed libraries in the system. It is typically used to compile and link against one or more libraries.
-------------------	--

10.24. Ncurses-6.0

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

10.24.1. Installation of Ncurses

Prepare Ncurses for compilation:

```
./configure \
  --prefix=/usr \
  --with-shared \
  --without-debug \
  --enable-widex \
  --enable-pc-files
```

The meaning of the new configure option:

--enable-pc-files

This tells Ncurses to generate and install .pc files for **pkg-config**.

Compile the package:

```
make
```

This package has a test suite, but it can only be run after the package is installed. The tests are in the `test /` directory. See the README file in that directory for details.

Install the package:

```
make install
```

Move the `libncursesw` shared library to `/lib` and create a new symlink in `/usr/lib`:

```
mv -v /usr/lib/libncursesw.so.* /lib
ln -svf ../../lib/$(readlink /usr/lib/libncursesw.so) /usr/lib/libncursesw.so
```

Many packages that use Ncurses will compile just fine against the widechar libraries, but won't know to look for them. Create linker scripts and symbolic links to allow older and non-widex compatible programs to build properly:

```
for lib in ncurses form panel menu ; do
  echo "INPUT(-l${lib}w)" > /usr/lib/lib${lib}.so
  ln -sfv lib${lib}w.a /usr/lib/lib${lib}.a
done
ln -sfv libncurses++w.a /usr/lib/libncurses++.a
ln -sfv ncursesw6-config /usr/bin/ncurses6-config
```

10.24.2. Contents of Ncurses

Installed programs:	captinfo (link to tic), clear, infocmp, infotocap (link to tic), ncursesw6-config, reset (link to tset), tabs, tic, toe, tput, tset
Installed libraries:	libcursesw.so (link to libncursesw.so), libformw.[a,so], libmenuw.[a,so], libncurses++w.a, libncursesw.[a,so], libpanelw.[a,so]
Installed directories:	/usr/share/tabset, /usr/share/terminfo

Short Descriptions

captoinfo	Converts a termcap description into a terminfo description
clear	Clears the screen, if possible
infocmp	Compares or prints out terminfo descriptions
infotocap	Converts a terminfo description into a termcap description
ncursesw6-config	Provides configuration information for ncurses
reset	Reinitializes a terminal to its default values
tabs	Sets and clears 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
libcursesw	A link to libncursesw
libncursesw	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
libformw	Contains functions to implement forms
libmenuw	Contains functions to implement menus
libpanelw	Contains functions to implement panels

10.25. Shadow-4.5

The Shadow package contains programs for handling passwords in a secure way.

10.25.1. Installation of Shadow

Note

If you would like to enforce the use of strong passwords, refer to <http://cblfs.clfs.org/index.php/Cracklib> for installing Cracklib prior to building Shadow. After Cracklib is installed, execute this **sed** in Shadow's source directory to correct the path to the Cracklib dictionary:

```
sed -i 's@\(\DICTPATH.\).*@\1/lib/cracklib/pw_dict@' etc/login.defs
```

Finally, add `--with-libcrack` to the **configure** command below.

Disable the installation of the **groups** program and man pages, as better versions of these programs are provided by Coreutils, Util-linux and Man-pages:

```
sed -i src/Makefile.in \
-e 's/groups$(EXEEXT) //' \
find man -name Makefile.in -exec sed -i \
-e 's/man1\/groups\.1 //' \
-e 's/man3\/getspnam\.3 //' \
-e 's/man5\passwd\.5 //' ' {} ' \;
```

Prepare Shadow for compilation:

```
./configure \
--sysconfdir=/etc \
--with-group-max-length=32
```

The meaning of the new configure option:

`--sysconfdir=/etc`

Tells Shadow to install its configuration files into `/etc`, rather than `/usr/etc`.

`--with-group-max-length=32`

The maximum user name is 32 characters. Make the maximum group name the same.

Compile the package:

```
make
```

This package does not come with a test suite.

Install the package:

```
make install
```

Instead of using the default *DES* method, use the more secure *SHA512* 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. Use the following sed command to make these changes to the appropriate configuration file:

```
sed -i /etc/login.defs \
    -e 's@#\ (ENCRYPT_METHOD \) .*@\1SHA512@' \
    -e 's@/var/spool/mail@/var/mail@'
```

Move a misplaced program to its proper location:

```
mv -v /usr/bin/passwd /bin
```

The **login** program will write to `/var/log/faillog`, to record failed login attempts, and `/var/log/lastlog`, to record the date and time of the latest successful login for each user. These log files are not created automatically if they do not already exist, so we will create them now and give them appropriate ownership and permissions:

```
touch /var/log/{fail,last}log
chgrp -v utmp /var/log/{fail,last}log
chmod -v 664 /var/log/{fail,last}log
```

10.25.2. 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
```

To view or change the default settings for new user accounts that you create, you can edit `/etc/default/useradd`. See **man useradd** or http://cblfs.clfs.org/index.php/Configuring_for_Adding_Users for more information.

10.25.3. Setting the root password

Choose a password for user `root` and set it by running:

```
passwd root
```

10.25.4. 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), 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
chgpaswd	Used to update group passwords in batch mode
chpaswd	Used to update the passwords of an entire series of user accounts
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
gpaswd	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 superuser privileges
groupmod	Is used to modify the given group's name or GID
grpck	Verifies the integrity of the group files <code>/etc/group</code> and <code>/etc/gshadow</code>
grpconv	Creates or updates the shadow group file from the normal group file
grpunconv	Updates <code>/etc/group</code> from <code>/etc/gshadow</code> 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. It is designed to be used as the default shell for disabled accounts.
passwd	Is used to change the password for a user or group account
pwck	Verifies the integrity of the password files <code>/etc/passwd</code> and <code>/etc/shadow</code>
pwconv	Creates or updates the shadow password file from the normal password file
pwunconv	Updates <code>/etc/passwd</code> from <code>/etc/shadow</code> 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 <code>/etc/group</code> or <code>/etc/gshadow</code> files
vipw	Edits the <code>/etc/passwd</code> or <code>/etc/shadow</code> files

10.26. Util-linux-2.29.2 Pass 1

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

10.26.1. Installation of Util-linux Pass 1

Prepare Util-linux for compilation:

```
./configure \
    ADJTIME_PATH=/var/lib/hwclock/adjtime \
    --enable-write \
    --disable-chfn-chsh \
    --disable-login \
    --disable-nologin \
    --disable-su \
    --disable-setpriv \
    --disable-runuser \
    --docdir=/usr/share/doc/util-linux-2.29.2
```

The meaning of the configure options:

--enable-write

This option allows the **write** program to be installed.

*--disable-**

This option disables various programs

Compile the package:

```
make
```

Install the package:

```
make install
```

Details on this package are located in Section 10.67.3, “Contents of Util-linux.”

10.27. Procps-ng-3.3.12

The Procps-ng package contains programs for monitoring processes.

10.27.1. Installation of Procps-ng

Prepare procps-ng for compilation:

```
./configure \
  --prefix=/usr \
  --exec-prefix= \
  --libdir=/usr/lib \
  --docdir=/usr/share/doc/procps-ng-3.3.12 \
  --disable-kill
```

The meaning of the configure options:

--disable-kill

This switch disables building the **kill** program - a better version was installed by the Util-linux package.

Compile the package:

```
make
```

Note

When using the boot method, two tests will fail if the hostname is not set. If you have booted the temporary system, and want to run the test suite, run the following command:

```
hostname clfs
```

If running the testsuite, first disable a test which fails when scripting does not use a tty device:

```
sed -i -r 's|(pmap_initname)\\$|\\1|' testsuite/pmap.test/pmap.exp
make check
```

Install the package:

```
make install
```

Move essential files to a location that can be found if /usr is not mounted:

```
mv -v /usr/lib/libprocps.so.* /lib
ln -sfv ../../lib/$(readlink /usr/lib/libprocps.so) /usr/lib/libprocps.so
```

10.27.2. Contents of Procps-ng

Installed programs:	free, pgrep, pidof, pkill, pmap, ps, pwdx, slabtop, sysctl, tload, top, uptime, vmstat, w, watch
Installed library:	libprocps.{a,so}
Installed directories:	/usr/include/proc, /usr/share/doc/procps-ng-3.3.12

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
pidof	Reports the PIDs of the given programs
kill	Signals processes based on their name and other attributes
map	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 slab cache information in real time
sysctl	Modifies kernel parameters at run time
load	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
libprocps	Contains the functions used by most programs in this package

10.28. E2fsprogs-1.43.4

The E2fsprogs package contains the utilities for handling the `ext2` file system. It also supports the `ext3` and `ext4` journaling file systems.

10.28.1. 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 \
  --bindir=/bin \
  --with-root-prefix="" \
  --enable-elf-shlibs \
  --disable-libblkid \
  --disable-libuuid \
  --disable-fsck \
  --disable-uidd
```

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 `uidd` daemon, and the **fsck** wrapper, as Util-Linux installed all of them earlier.

Compile the package:

```
make
```

To test the results, issue:

```
make check
```

Install the binaries, documentation and shared libraries:

```
make install
```

Install the static libraries and headers:

```
make install-libs
```

10.28.2. 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, tune2fs
Installed libraries:	libcom_err.[a,so], libe2p.[a,so], libext2fs.[a,so], libss.[a,so], libquota.a
Installed directories:	/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 on a Linux file system
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 <code>com_err</code> library
debugfs	A file system debugger; it can be used to examine and change the state of an <code>ext2</code> 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 <code>ext2</code> , <code>ext3</code> and <code>ext4</code> file systems
e2image	Is used to save critical <code>ext2</code> 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 <code>ext2</code> file system present on a given device
e2undo	Replays an undo log for an <code>ext2/ext3/ext4</code> filesystem
e4defrag	Online defragmenter for <code>ext4</code> filesystems
filefrag	Reports on how badly fragmented a particular file might be
fsck.ext2	By default checks <code>ext2</code> file systems
fsck.ext3	By default checks <code>ext3</code> file systems
fsck.ext4	By default checks <code>ext4</code> file systems
fsck.ext4dev	By default checks <code>ext4dev</code> file systems
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 <code>libss</code> subsystem library
mke2fs	Creates an <code>ext2</code> , <code>ext3</code> or <code>ext4</code> file system on the given device
mkfs.ext2	By default creates <code>ext2</code> file systems
mkfs.ext3	By default creates <code>ext3</code> file systems
mkfs.ext4	By default creates <code>ext4</code> file systems
mkfs.ext4dev	By default creates <code>ext4dev</code> file systems

mklost+found	Used to create a <code>lost+found</code> directory on an <code>ext2</code> 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 <code>ext2</code> file system
tune2fs	Adjusts tunable file system parameters on an <code>ext2</code> file system
<code>libcom_err</code>	The common error display routine
<code>libe2p</code>	Used by dumpe2fs , chattr , and lsattr
<code>libext2fs</code>	Contains routines to enable user-level programs to manipulate an <code>ext2</code> file system
<code>libquota</code>	Provides an interface for creating and updating quota files and <code>ext4</code> superblock fields
<code>libss</code>	Used by debugfs

10.29. Coreutils-8.27

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

10.29.1. Installation of Coreutils

A known issue with the **uname** program from this package is that the **-p** switch always returns unknown. The following patch fixes this behavior for all architectures:

```
patch -Np1 -i ../coreutils-8.27-uname-1.patch
```

Now prepare Coreutils for compilation:

```
FORCE_UNSAFE_CONFIGURE=1 \
./configure \
  --prefix=/usr \
  --enable-no-install-program=kill,uptime \
  --enable-install-program=hostname \
  --libexecdir=/usr/lib
```

The meaning of the configure options:

```
FORCE_UNSAFE_CONFIGURE=1
```

Forces Coreutils to compile when using the root user.

Compile the package:

```
make
```

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
```

The test suite will now be run as the **nobody** user. Some tests require that the user be a member of more than one group. Add a temporary group and make the user **nobody** a part of it so that the tests are not skipped:

```
echo "dummy:x:1000:nobody" >> /etc/group
```

Fix permissions of some files so the non-root user can compile and run the tests:

```
chown -Rv nobody .
```

Then run the remainder of the tests as the **nobody** user:

```
su nobody -s /bin/bash \
  -c "PATH=$PATH make RUN_EXPENSIVE_TESTS=yes -k check || true"
```

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} /bin
mv -v /usr/bin/{dd,df,echo,false,hostname,ln,ls,mkdir,mknod} /bin
mv -v /usr/bin/{mv,pwd,rm,rmdir,stat,tee,true,uname} /bin
mv -v /usr/bin/chroot /usr/sbin
```

10.29.2. 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, hostname, id, install, join, link, ln, logname, ls, md5sum, mkdir, mkfifo, mknod, mktemp, mv, nice, nl, nohup, nproc, numfmt, od, paste, pathchk, pinky, pr, printenv, printf, ptx, pwd, readlink, realpath, rm, rmdir, runcon, seq, sha1sum, sha224sum, sha256sum, sha384sum, sha512sum, shred, shuf, sleep, sort, split, stat, stdbuf, stty, sum, sync, tac, tail, tee, test, timeout, touch, tr, true, truncate, tsort, tty, uname, unexpand, uniq, unlink, users, vdir, wc, who, whoami, yes

Installed library: libstdbuf.so

Installed directory: /usr/lib/coreutils

Short Descriptions

base64	Base64 encode/decode data and print to standard output
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
hostname	Reports or sets the name 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 the current process
numfmt	Converts numbers to or from human-readable strings
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 SHA224 checksums
sha256sum	Prints or checks SHA256 checksums
sha384sum	Prints or checks SHA384 checksums
sha512sum	Prints or checks SHA512 checksums
shred	Overwrites the given files repeatedly with complex patterns, making it difficult to recover the data
shuf	Write a random permutation of the input lines to standard output or a file
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 a command with modified 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 or [Compares values and checks file types
timeout	Runs a command with a time limit
touch	Changes file timestamps, setting the access and modification times of the given files to the current time; files that do not exist are created with zero length

tr	Translates, squeezes, and deletes the given characters from standard input
true	Does nothing, successfully; it always exits with a status code indicating success
truncate	Shrinks or expands a file to the specified size
tsort	Performs a topological sort; it writes a completely ordered list according to the partial ordering in a given file
tty	Reports the file name of the terminal connected to standard input
uname	Reports system information
unexpand	Converts spaces to tabs
uniq	Discards all but one of successive identical lines
unlink	Removes the given file
users	Reports the names of the users currently logged on
vdir	Is the same as ls -l
wc	Reports the number of lines, words, and bytes for each given file, as well as a total line when more than one file is given
who	Reports who is logged on
whoami	Reports the user name associated with the current effective user ID
yes	Repeatedly outputs “y” or a given string until killed
libstdbuf	Library used by stdbuf

10.30. Iana-Etc-2.30

The Iana-Etc package provides data for network services and protocols.

10.30.1. Installation of Iana-Etc

The following patch contains xml files which provide updates to the services and protocol files:

```
xzcat ../iana-etc-2.30-numbers_update-20140202-2.patch.xz | patch -Np1 -i -
```

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
```

10.30.2. Contents of Iana-Etc

Installed files: `/etc/protocols`, `/etc/services`

Short Descriptions

<code>/etc/protocols</code>	Describes the various DARPA Internet protocols that are available from the TCP/IP subsystem
<code>/etc/services</code>	Provides a mapping between friendly textual names for internet services, and their underlying assigned port numbers and protocol types

10.31. Libtool-2.4.6

The Libtool package contains the GNU generic library support script. It wraps the complexity of using shared libraries in a consistent, portable interface.

10.31.1. Installation of Libtool

Prepare Libtool for compilation:

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

Compile the package:

```
make
```

To test the results, issue:

```
make check
```

Install the package:

```
make install
```

10.31.2. Contents of Libtool

Installed programs:	libtool, 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

10.32. IPRoute2-4.9.0

The IPRoute2 package contains programs for basic and advanced IPV4-based networking.

10.32.1. Installation of IPRoute2

ARPD will not be installed as Berkeley DB is not installed. Remove any **arpd** references during install.

```
sed -i '/ARPD/d' Makefile
      sed -i 's/arpd.8//' man/man8/Makefile
      sed -i '/tc-simple/s@tc-skbmod.8 @@' man/man8/Makefile
      rm -v doc/arpd.sgml
```

Compile the package:

```
make
```

This package does not come with a test suite.

Install the package:

```
make DOCDIR=/usr/share/doc/iproute2-4.9.0 install
```

10.32.2. Contents of IPRoute2

Installed programs:	bridge, ctstat (link to lnstat), genl, ifcfg, ifstat, ip, lnstat, nstat, route, routef, route, rtacct, rtmon, rtpr, rtstat (link to lnstat), ss, tc
Installed directories:	/etc/iproute2, /usr/lib/tc, /usr/share/doc/iproute2-4.9.0

Short Descriptions

bridge	Configures network bridges
ctstat	Connection status utility
genl	Needs description
ifcfg	A shell script wrapper for the ip command
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 <code>/proc/net/route</code>
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 <ul style="list-style-type: none"> tc qdisc allows users to setup the queueing discipline tc class allows users to setup classes based on the queueing 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

10.33. 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**.

10.33.1. Installation of Bzip2

By default Bzip2 creates some symlinks that use absolute pathnames. The following sed will cause them to be created with relative paths instead:

```
sed -i -e 's:ln -s -f $(PREFIX)/bin:ln -s :' Makefile
```

Make Bzip2 install its manpages in `/usr/share/man` instead of `/usr/man`:

```
sed -i 's@X)/man@X)/share/man@g' ./Makefile
```

The Bzip2 package does not contain a **configure** script. Compile it with:

```
make -f Makefile-libbz2_so
make clean
```

The `-f` flag 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.

Recompile the package using a non-shared library and test it:

```
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
```

10.33.2. Contents of Bzip2

Installed programs:	bunzip2 (link to bzip2), bzcat (link to bzip2), bzcmp (link to bzdiff), bzdiff, bzegrep (link to bzgrep), bzfgrep (link to bzgrep), bzgrep, bzip2, bzip2recover, bzless (link to bzmored), bzmored
Installed libraries:	libbz2.a, libbz2.so (link to libbz2.so.1.0), libbz2.so.1.0 (link to libbz2.so.1.0.6), libbz2.so.1.0.6

Short Descriptions

bunzip2	Decompresses bziped files
bzcat	Decompresses to standard output

bzcmp	Runs cmp on bziped files
bzdiff	Runs diff on bziped files
bzegrep	Runs egrep on bziped files
bzfgrep	Runs fgrep on bziped files
bzgrep	Runs grep on bziped 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 bziped files
bzless	Runs less on bziped files
bzmore	Runs more on bziped files
libbz2*	The library implementing lossless, block-sorting data compression, using the Burrows-Wheeler algorithm

10.34. GDBM-1.13

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.

10.34.1. 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 CLFS may require the older DBM routines it provides.

Compile the package:

```
make
```

To test the results, issue:

```
make check
```

Install the package:

```
make install
```

10.34.2. Contents of GDBM

Installed programs:	<code>gdbm_dump</code> , <code>gdbm_load</code> , <code>gdbmtool</code>
Installed libraries:	<code>libgdbm.{a,so}</code> , <code>libgdbm_compat.{a,so}</code>

Short Descriptions

gdbm_dump	Dumps a GDBM database to a file.
gdbm_load	Recreates a GDBM database from a dump file.
gdbmtool	Tests and modifies a GDBM database
libgdbm	Contains functions to manipulate a hashed database
libgdbm_compat	Compatibility library containing older DBM functions

10.35. Perl-5.26.0

The Perl package contains the Practical Extraction and Report Language.

10.35.1. Installation of Perl

By default, Perl's `Compress::Raw::Zlib` and `Compress::Raw::Bzip2` modules build and link against internal copies of Zlib and Bzip2. The following command will make Perl use the system-installed copies of these libraries:

```
export BUILD_ZLIB=False
export BUILD_BZIP2=0
```

Note

If you are following the boot method you will need to enable the loopback device:

```
ip link set lo up
```

Before starting to configure, create a basic `/etc/hosts` file which will be referenced by one of Perl's configuration files as well as used by the test suite:

```
echo "127.0.0.1 localhost $(hostname)" > /etc/hosts
```

To have full control over the way Perl is set up, you can run the interactive **Configure** script and hand-pick the way this package is built. If you prefer instead to use the defaults that Perl auto-detects, prepare Perl for compilation with:

```
./configure.gnu \
  --prefix=/usr \
  -Dvendorprefix=/usr \
  -Dman1dir=/usr/share/man/man1 \
  -Dman3dir=/usr/share/man/man3 \
  -Dpager="/bin/less -isR" \
  -Dusethreads \
  -Duseshrplib
```

The meaning of the configure option:

`-Dman1dir=/usr/share/man/man1 -Dman3dir=/usr/share/man/man3`

Since Groff is not installed yet, **configure.gnu** thinks that we do not want man pages for Perl. Issuing these parameters overrides this decision.

`-Dpager="/bin/less -isR"`

Less has not yet been installed, so by default **perldoc** will invoke the **more** program for viewing documentation. This option ensures that it will use **less** instead.

`-Dusethreads`

This tells Perl to use threads.

`-Duseshrplib`

This tells Perl to build a shared libperl.

Compile the package:

```
make
```

To test the results, issue:

```
make test
```

Install the package and remove the variables set previously:

```
make install
unset BUILD_ZLIB BUILD_BZIP2
```

10.35.2. 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, perl5.26.0 (link to perl), perlbug, perldoc, perlvp, 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, 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	Shell script that provides a command interface to CPAN.pm
cpan2dist	The CPANPLUS distribution creator
cpanp	The CPANPLUS launcher
cpanp-run-perl	Perl script that (description needed)
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	A 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	Combines some of the best features of C, sed , awk and sh into a single swiss-army-knife language
perl5.26.0	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	A command-line tool for running tests against Test::Harness
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 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

10.36. Readline-7.0

The Readline package is a set of libraries that offers command-line editing and history capabilities.

10.36.1. Installation of Readline

The following patch contains updates from the maintainer. The maintainer of Readline only releases these patches to fix serious issues:

```
patch -Np1 -i ../readline-7.0-branch_update-1.patch
```

Reinstalling Readline moves the old libraries to <libraryname> and a linking bug may occur in **ldconfig**. Prevent this with the following seds:

```
sed -i '/MV.*old/d' Makefile.in
sed -i '/{OLDSUFF}/c:' support/shlib-install
```

Prepare Readline for compilation:

```
./configure \
  --prefix=/usr \
  --libdir=/lib \
  --docdir=/usr/share/doc/readline-7.0
```

Compile the package:

```
make SHLIB_LIBS=-lncurses
```

This package does not come with a test suite.

Install the package:

```
make SHLIB_LIBS=-lncurses htmdir=/usr/share/doc/readline-7.0 install
```

Now move the static libraries to a more appropriate location:

```
mv -v /lib/lib{readline,history}.a /usr/lib
```

Next, relink the dynamic libraries into /usr/lib and remove the .so files in /lib.

```
ln -svf ../../lib/${readlink /lib/libreadline.so} /usr/lib/libreadline.so
ln -svf ../../lib/${readlink /lib/libhistory.so} /usr/lib/libhistory.so
rm -v /lib/lib{readline,history}.so
```

10.36.2. Contents of Readline

Installed libraries:	libhistory.[a,so], libreadline.[a,so]
Installed directories:	/usr/include/readline, /usr/share/doc/readline-7.0, /usr/share/readline

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

10.37. Autoconf-2.69

The Autoconf package contains programs for producing shell scripts that can automatically configure source code.

10.37.1. Installation of Autoconf

Prepare Autoconf for compilation:

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

Compile the package:

```
make
```

To test the results, issue:

```
make check VERBOSE=yes
```

17 tests are skipped that use Automake and different GCC languages. For full test coverage, Autoconf can be re-tested after Automake has been installed.

Install the package:

```
make install
```

10.37.2. Contents of Autoconf

Installed programs:	autoconf, autoheader, autom4te, autoreconf, autoscan, autoupdate, 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 <i>#define</i> 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 <code>configure.in</code> file for a software package; it examines the source files in a directory tree, searching them for common portability issues, and creates a <code>configure.scan</code> file that serves as a preliminary <code>configure.in</code> file for the package
autoupdate	Modifies a <code>configure.in</code> file that still calls autoconf macros by their old names to use the current macro names
ifnames	Helps when writing <code>configure.in</code> 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 <code>configure.in</code> file generated by autoscan

10.38. Automake-1.15

The Automake package contains programs for generating Makefiles for use with Autoconf.

10.38.1. Installation of Automake

Apply the following patch to fix outdated syntax that is no longer recognized by Perl-5.26.0.

```
patch -Np1 -i ../automake-1.15-perl_5_26-1.patch
```

Prepare Automake for compilation:

```
./configure \
  --prefix=/usr \
  --docdir=/usr/share/doc/automake-1.15
```

Compile the package:

```
make
```

To test the results, issue:

```
make check
```

Install the package:

```
make install
```

10.38.2. Contents of Automake

Installed programs:	aclocal, aclocal-1.15, automake, automake-1.15, compile, config.guess, config.sub, depcomp, install-sh, mdate-sh, missing, mkinstalldirs, py-compile, symlink-tree, ylwrap
Installed directories:	/usr/share/aclocal-1.15, /usr/share/automake-1.15, /usr/share/doc/automake

Short Descriptions

aclocal	Generates <code>aclocal.m4</code> files based on the contents of <code>configure.in</code> files
aclocal-1.15	A hard link to aclocal
automake	A tool for automatically generating <code>Makefile.in</code> files from <code>Makefile.am</code> files. To create all the <code>Makefile.in</code> files for a package, run this program in the top-level directory. By scanning the <code>configure.in</code> file, it automatically finds each appropriate <code>Makefile.am</code> file and generates the corresponding <code>Makefile.in</code> file
automake-1.15	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

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

10.39. Bash-4.4

The Bash package contains the Bourne-Again SHell.

10.39.1. Installation of Bash

The following patch contains updates from the maintainer. The maintainer of Bash only releases these patches to fix serious issues:

```
patch -Np1 -i ../bash-4.4-branch_update-1.patch
```

Prepare Bash for compilation:

```
./configure \
  --prefix=/usr \
  --without-bash-malloc \
  --with-installed-readline \
  --docdir=/usr/share/doc/bash-4.4
```

The meaning of the new configure option:

--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
```

To test the results, issue:

```
make tests
```

Install the package:

```
make install
```

Move the **bash** binary to `/bin`, overwriting the symlink that was previously created:

```
mv -v /usr/bin/bash /bin
```

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.

10.39.2. Contents of Bash

Installed programs: `bash`, `bashbug`, `sh` (link to `bash`)
Installed directory: `/usr/share/doc/bash-4.4`

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

10.40. Bc-1.07.1

The Bc package contains an arbitrary precision numeric processing language.

10.40.1. Installation of Bc

Change an internal script to use **sed** instead of **ed**:

```
cat > bc/fix-libmath_h << "EOF"
#! /bin/bash
sed -e '1      s/^\{"/' \
    -e      's/$/"/' \
    -e '2,$ s/^\{"/' \
    -e '$ d' \
    -i libmath.h

sed -e '$ s/$/0}/' \
-i libmath.h
EOF
```

Prepare Bc for compilation:

```
./configure \
  --prefix=/usr \
  --with-readline \
  --mandir=/usr/share/man \
  --infodir=/usr/share/info
```

Compile the package:

```
make
```

To test the results, issue:

```
echo "quit" | ./bc/bc -l Test/checklib.b
```

Install the package:

```
make install
```

10.40.2. Contents of Bc

Installed programs: bc, dc

Short Descriptions

bc is a command line calculator

dc is a reverse-polish command line calculator

10.41. Diffutils-3.6

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

10.41.1. Installation of Diffutils

Fix a bug that prevents locale files from being installed:

```
sed -i 's:= @mkdir_p@:= /bin/mkdir -p:' po/Makefile.in.in
```

Prepare Diffutils for compilation:

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

Diffutils wants **ed** as the default editor for **sdiff**. The following **sed** will change the default to **vi**:

```
sed -i 's@\(^\#define DEFAULT_EDITOR_PROGRAM \).*@\1"vi"@' lib/config.h
```

Compile the package:

```
make
```

To test the results, issue:

```
make check
```

Install the package:

```
make install
```

10.41.2. Contents of Diffutils

Installed programs: cmp, diff, diff3, 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

10.42. File-5.31

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

10.42.1. 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
```

10.42.2. 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

10.43. Gawk-4.1.4

The Gawk package contains programs for manipulating text files.

10.43.1. 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
```

Install the documentation:

```
mkdir -v /usr/share/doc/gawk-4.1.4
cp -v doc/{awkforai.txt,*.eps,pdf,jpg} /usr/share/doc/gawk-4.1.4
```

10.43.2. Contents of Gawk

Installed programs: awk (link to gawk), gawk, gawk-4.1.4, grcat, igawk, pgawk, pgawk-4.1.4, pwcat

Installed directories: /usr/lib/awk, /usr/lib/gawk, /usr/share/awk, /usr/share/doc/gawk-4.1.4

Short Descriptions

awk	A link to gawk
gawk	A program for manipulating text files; it is the GNU implementation of awk
gawk-4.1.4	A hard link to gawk
grcat	Dumps the group database <code>/etc/group</code>
igawk	Gives gawk the ability to include files
pgawk	The profiling version of gawk
pgawk-4.1.4	Hard link to pgawk
pwcat	Dumps the password database <code>/etc/passwd</code>

10.44. Findutils-4.6.0

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).

10.44.1. Installation of Findutils

Prepare Findutils for compilation:

```
./configure \
  --prefix=/usr \
  --libexecdir=/usr/lib/locate \
  --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
```

10.44.2. Contents of Findutils

Installed programs:	bigram, code, find, frcode, locate, oldfind, updatedb, xargs
Installed directory:	/usr/lib/locate

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

10.45. Gettext-0.19.8.1

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

10.45.1. Installation of Gettext

Prepare Gettext for compilation:

```
./configure \
  --prefix=/usr \
  --docdir=/usr/share/doc/gettext-0.19.8.1
```

Compile the package:

```
make
```

To test the results, issue:

```
make check
```

Install the package:

```
make install
```

10.45.2. 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, xgettext
Installed libraries:	libasprintf.[a,so], libgettextlib.so, libgettextpo.[a,so], libgettextsrc.so, preloadable_libintl.so
Installed directories:	/usr/lib/gettext, /usr/share/doc/gettext-0.19.8.1, /usr/share/gettext

Short Descriptions

autopoint	Copies standard Gettext infrastructure files into a source package
config.charset	Outputs a system-dependent table of character encoding aliases
config.rpath	Outputs a system-dependent set of variables, describing how to set the runtime search path of shared libraries in an executable
envsubst	Substitutes environment variables in shell format strings
gettext	Translates a natural language message into the user's language by looking up the translation in a message catalog
gettext.sh	Primarily serves as a shell function library for gettext
gettextize	Copies all standard Gettext files into the given top-level directory of a package to begin internationalizing it
hostname	Displays a network hostname in various forms

msgattrib	Filters the messages of a translation catalog according to their attributes and manipulates the attributes
msgcat	Concatenates and merges the given .po files
msgcmp	Compares two .po files to check that both contain the same set of msgid strings
msgcomm	Finds the messages that are common to 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	Recode 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 <i>autosprintf</i> class, which makes C formatted output routines usable in C++ programs, for use with the <i><string></i> strings and the <i><iostream></i> 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.so	A library, intended to be used by LD_PRELOAD, that assists libintl in logging untranslated messages.

10.46. Gperf-3.0.4

Gperf generates a perfect hash function from a key set.

10.46.1. Installation of Gperf

Prepare Gperf for compilation:

```
./configure \  
  --prefix=/usr \  
  --docdir=/usr/share/doc/gperf-3.0.4
```

Compile the package:

```
make
```

To test the results, issue:

```
make check
```

Install the package:

```
make install
```

10.46.2. Contents of Gperf

Installed program:	gperf
Installed directory:	/usr/share/doc/gperf-3.0.4

Short Descriptions

gperf Generates a perfect hash function from a key set

10.47. Grep-3.0

The Grep package contains programs for searching through files.

10.47.1. 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
```

10.47.2. Contents of Grep

Installed programs: egrep, fgrep, 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

10.48. Groff-1.22.3

The Groff package contains programs for processing and formatting text.

10.48.1. 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.

Prepare Groff for compilation:

```
PAGE=[paper_size] ./configure \
--prefix=/usr
```

Compile the package:

```
make
```

This package does not come with a test suite.

Install the package:

```
make install
```

10.48.2. Contents of Groff

Installed programs:	addftinfo, afmtodit, chem, eqn, eqn2graph, gdiffmk, grap2graph, grn, grodvi, groff, groffer, grog, grolbp, grolj4, grops, grotty, hpftodit, indxbib, lkbib, lookbib, mmroff, neqn, nroff, pdfroff, pfbtops, pic, pic2graph, post-grohtml, pre-grohtml, preconv, refer, roff2dvi, roff2html, roff2pdf, roff2ps, roff2text, roff2x, soelim, tbl, tfmtodit, troff
Installed directories:	/usr/lib/groff, /usr/share/doc/groff-1.22.3, /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
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 <code>-e</code> , <code>-man</code> , <code>-me</code> , <code>-mm</code> , <code>-ms</code> , <code>-p</code> , <code>-s</code> , and <code>-t</code> 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
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 <code>.pfb</code> 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
pre-grohtml	Translates the output of GNU troff to HTML
preconv	Converts encoding of input files to something GNU troff understands
refer	Copies the contents of a file to the standard output, except that lines between <code>./</code> and <code>./</code> are interpreted as citations, and lines between <code>.R1</code> and <code>.R2</code> are interpreted as commands for how citations are to be processed
roff2dvi	Transforms roff files into other formats
roff2html	Transforms roff files into other formats
roff2pdf	Transforms roff files into other formats
roff2ps	Transforms roff files into other formats
roff2text	Transforms roff files into other formats
roff2x	Transforms roff files into other formats
soelim	Reads files and replaces lines of the form <code>.so file</code> by the contents of the mentioned <i>file</i>
tbl	Compiles descriptions of tables embedded within troff input files into commands that are understood by troff

tfmtofit

Creates a font file for use with **groff -Tdv**

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

10.49. Less-491

The Less package contains a text file viewer.

10.49.1. Installation of Less

Prepare Less for compilation:

```
./configure \
  --prefix=/usr \
  --sysconfdir=/etc
```

Compile the package:

```
make
```

This package does not come with a test suite.

Install the package:

```
make install
```

Move **less** to `/bin`:

```
mv -v /usr/bin/less /bin
```

10.49.2. Contents of Less

Installed programs: less, lessecho, 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 <code>*</code> and <code>?</code> , in filenames on Unix systems
lesskey	Used to specify the key bindings for less

10.50. Gzip-1.8

The Gzip package contains programs for compressing and decompressing files.

10.50.1. 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
```

Now we will move some of the utilities to `/usr/bin` to meet FHS compliance:

```
mv -v /bin/{gzexe,uncompress} /usr/bin
mv -v /bin/z{egrep,cmp,diff,fgrep,force,grep,less,more,new} /usr/bin
```

10.50.2. Contents of Gzip

Installed programs: gunzip, gzexe, gzip, uncompress, zcat, zcmp, zdiff, zegrep, zfgrep, zforce, zgrep, zless, zmore, 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 <code>.gz</code> 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

10.51. IPutils-s20150815

The IPutils package contains programs for basic networking.

10.51.1. Installation of IPutils

Apply the following patch for man-pages and build edits:

```
patch -Np1 -i ../iputils-s20150815-build-1.patch
```

Compile the package:

```
make TARGETS="clockdiff ping rdisc tracepath tracepath6 traceroute6"
```

This package does not come with a test suite.

Install the package:

```
install -v -m755 ping /bin
install -v -m755 clockdiff /usr/bin
install -v -m755 rdisc /usr/bin
install -v -m755 tracepath /usr/bin
install -v -m755 trace{path,route}6 /usr/bin
install -v -m644 doc/*.8 /usr/share/man/man8
ln -sv ping /bin/ping4
ln -sv ping /bin/ping6
```

10.51.2. Contents of iputils

Installed programs: clockdiff, ping, rdisc, tracepath, tracepath6, traceroute6, ping4 (link to ping), ping6 (link to ping)

Short Descriptions

clockdiff	Measures the clock difference between hosts
ping	Sends echo-request packets and reports how long the replies take.
ping	Executes ping for IPV4 support
ping6	Executes ping for IPV6 support
rdisc	Network router discovery daemon
tracepath	Traces the path to a network host discovering MTU along the path. This is the IPV4 version.
tracepath6	Traces the path to a network host discovering MTU along the path. This is the IPV6 version.
traceroute6	Traces the path to a network host on an IPV6 network

10.52. Kbd-2.0.4

The Kbd package contains key-table files and keyboard utilities.

10.52.1. Installation of Kbd

Prepare Kbd for compilation:

```
PKG_CONFIG_PATH="/tools/lib/pkgconfig" \
./configure \
  --prefix=/usr \
  --disable-vlock \
  --enable-optional-progs
```

The meaning of the new configure options:

`PKG_CONFIG_PATH`

Use pkg-config to obtain the location of the test library metadata built in Section 6.13, “Check-0.11.0”.

`--disable-vlock`

Prevents Kbd from trying to build the **vlock** program, which requires Linux-PAM.

`--enable-optional-progs`

Installs several additional programs.

Compile the package:

```
make
```

To test the results, issue:

```
make check
```

Install the package:

```
make install
```

Some of the programs from Kbd are used by systemd to initialize the system, so those binaries need to be on the root partition:

```
mv -v /usr/bin/{loadkeys,setfont} /bin
```

Install the documentation:

```
mkdir -v /usr/share/doc/kbd-2.0.4
cp -R -v docs/doc/* /usr/share/doc/kbd-2.0.4
```

10.52.2. Contents of Kbd

Installed programs:

chvt, deallocvt, dumpkeys, fgconsole, getkeycodes, kbinfo, kbd_mode, kbdrate, loadkeys, loadunimap, mapscrn, openvt, psfaddtable (link to psfxtable), psfgettable (link to psfxtable), psfstriptime (link to psfxtable), psfxtable, resizecons, setfont, setkeycodes, setleds, setmetamode, setvtrgb, showconsolefont, showkey, unicode_start, unicode_stop

Installed directories:

/usr/share/consolefonts, /usr/share/consoletrans, /usr/share/doc/kbd-2.0.4, /usr/share/keymaps, /usr/share/unimaps

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 console
kbd_mode	Reports or sets the keyboard mode
kbdrate	Sets the keyboard repeat and delay rates
loadkeys	Loads the keyboard translation tables
loadunimap	Loads the kernel unicode-to-font mapping table
mapscrn	An obsolete program that used to load a user-defined output character mapping table into the console driver; this is now done by setfont
openvt	Starts a program on a new virtual terminal (VT)
psfaddtable	Adds a Unicode character table to a console font
psfgettable	Extracts the embedded Unicode character table from a console font
psfstriptime	Removes the embedded Unicode character table from a console font
psfxtable	Handle Unicode character tables for console fonts
resizecons	Changes the kernel idea of the console size
setfont	Changes the Enhanced Graphic Adapter (EGA) and Video Graphics Array (VGA) fonts on the console
setkeycodes	Loads kernel scancode-to-keycode mapping table entries; this is useful if there are unusual keys on the keyboard
setleds	Sets the keyboard flags and Light Emitting Diodes (LEDs)
setmetamode	Defines the keyboard meta-key handling
setvtrgb	Sets the virtual terminal RGB colors
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. Never use it on CLFS, because applications are not configured to support UNICODE.
unicode_stop	Reverts keyboard and console from UNICODE mode

10.53. Libpipeline-1.4.1

The Libpipeline package contains a library for manipulating pipelines of subprocesses in a flexible and convenient way.

10.53.1. Installation of Libpipeline

Prepare Libpipeline for compilation:

```
PKG_CONFIG_PATH=/tools/lib/pkgconfig \  
./configure \  
--prefix=/usr
```

Compile the package:

```
make
```

To test the results, issue:

```
make check
```

Install the package:

```
make install
```

10.53.2. Contents of Libpipeline

Installed libraries: libpipeline.so

Short Descriptions

`libpipeline` This library is used to safely construct pipeline between subprocesses

10.54. Man-DB-2.7.6.1

The Man-DB package contains programs for finding and viewing man pages.

10.54.1. Installation of Man-DB

Prepare Man-DB for compilation:

```
./configure \
  --prefix=/usr \
  --libexecdir=/usr/lib \
  --docdir=/usr/share/doc/man-db-2.7.6.1 \
  --sysconfdir=/etc \
  --disable-setuid \
  --enable-cache-owner=bin \
  --with-browser=/usr/bin/lynx \
  --with-vgrind=/usr/bin/vgrind \
  --with-grap=/usr/bin/grap
```

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 CBLFS 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 CLFS or CBLFS, but you should be able to install them yourself after finishing CLFS if you wish to do so.

Compile the package:

```
make
```

To test the results, issue:

```
make check
```

Install the package:

```
make install
```

Edit the installed tmpfile as we have no man user:

```
sed -i "s:man man:root root:g" /usr/lib/tmpfiles.d/man-db.conf
```

10.54.2. Non-English Manual Pages in CLFS

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 10.1. Expected character encoding of legacy 8-bit manual pages

Language (code)	Encoding	Language (code)	Encoding
Danish (da)	ISO-8859-1	Croatian (hr)	ISO-8859-2
German (de)	ISO-8859-1	Hungarian (hu)	ISO-8859-2
English (en)	ISO-8859-1	Japanese (ja)	EUC-JP
Spanish (es)	ISO-8859-1	Korean (ko)	EUC-KR
Estonian (et)	ISO-8859-1	Lithuanian (lt)	ISO-8859-13
Finnish (fi)	ISO-8859-1	Latvian (lv)	ISO-8859-13
French (fr)	ISO-8859-1	Macedonian (mk)	ISO-8859-5
Irish (ga)	ISO-8859-1	Polish (pl)	ISO-8859-2
Galician (gl)	ISO-8859-1	Romanian (ro)	ISO-8859-2
Indonesian (id)	ISO-8859-1	Russian (ru)	KOI8-R
Icelandic (is)	ISO-8859-1	Slovak (sk)	ISO-8859-2
Italian (it)	ISO-8859-1	Slovenian (sl)	ISO-8859-2
Norwegian Bokmal (nb)	ISO-8859-1	Serbian Latin (sr@latin)	ISO-8859-2
Dutch (nl)	ISO-8859-1	Serbian (sr)	ISO-8859-5
Norwegian Nynorsk (nn)	ISO-8859-1	Turkish (tr)	ISO-8859-9
Norwegian (no)	ISO-8859-1	Ukrainian (uk)	KOI8-U
Portuguese (pt)	ISO-8859-1	Vietnamese (vi)	TCVN5712-1
Swedish (sv)	ISO-8859-1	Simplified Chinese (zh_CN)	GBK
Belarusian (be)	CP1251	Simplified Chinese, Singapore (zh_SG)	GBK
Bulgarian (bg)	CP1251	Traditional Chinese, Hong Kong (zh_HK)	BIG5HKSCS
Czech (cs)	ISO-8859-2	Traditional Chinese (zh_TW)	BIG5
Greek (el)	ISO-8859-7		

Note

Manual pages in languages not in the list are not supported.

10.54.3. Contents of Man-DB

Installed programs: accessdb, apropos (link to whatis), catman, lexicog, man, mandb, manpath, whatis, zsoelim

Installed libraries: libman.so, libmandb.so

Installed directories: /usr/lib/man-db, /usr/share/doc/man-db-2.7.6.1

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 <i>.so file</i> by the contents of the mentioned <i>file</i>
libman	Contains run-time support for man
libmandb	Contains run-time support for man

10.55. Make-4.2.1

The Make package contains a program for compiling packages.

10.55.1. Installation of Make

Prepare Make for compilation:

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

Compile the package:

```
make
```

To test the results, issue:

```
make check
```

Install the package:

```
make install
```

10.55.2. 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

10.56. XZ Utils-5.2.3

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

10.56.1. Installation of XZ Utils

Prepare XZ Utils for compilation:

```
./configure \
  --prefix=/usr \
  --docdir=/usr/share/doc/xz-5.2.3
```

Compile the package:

```
make
```

To test the results, issue:

```
make check
```

Install the programs:

```
make install
```

Move the xz binary, and several symlinks that point to it, into the /bin directory:

```
mv -v /usr/bin/{xz,lzma,lzcat,unlzma,unxz,xzcat} /bin
```

Finally, move the shared library to a more appropriate location, and recreate the symlink pointing to it:

```
mv -v /usr/lib/liblzma.so.* /lib
ln -sfv ../../lib/$(readlink /usr/lib/liblzma.so) /usr/lib/liblzma.so
```

10.56.2. Contents of XZ Utils

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, /usr/share/doc/xz-5.2.3

Short Descriptions

lzcat	Decompresses LZMA and xz files
lzcmp	Compares lzma compressed files
lzdiff	Compares lzma compressed files
lzegrep	Runs egrep on lzma compressed files
lzfgrep	Runs fgrep on lzma compressed files

lzgrep	Runs grep on lzma compressed files
lzless	Runs less on lzma files
lzma	Compresses lzma files
lzmadec	Decompresses lzma files
lzmainfo	Displays information stored in an .lzma file header
lzmore	Runs more on lzma files
unlzma	Uncompresses lzma files
unxz	Uncompresses xz files
xz	Creates xz compressed files
xzcat	Decompresses xz files
xzcmp	Compares xz compressed files
xzdec	Decompresses to standard output
xzdiff	Compares xz compressed files
xzegrep	Runs egrep on xz compressed files
xzfgrep	Runs fgrep on xz compressed files
xzgrep	Runs grep on xz compressed files
xzless	Runs less on xz files
xzmore	Runs more on xz files
liblzma	The LZMA library

10.57. Expat-2.2.0

Expat is a stream-oriented XML parser library written in C.

10.57.1. Installation of Expat

Prepare Expat for compilation:

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

Compile the package:

```
make
```

To test the results, issue:

```
make check
```

Install the package:

```
make install
```

Install the documentation:

```
install -v -m755 -d /usr/share/doc/expat-2.2.0
install -v -m644 doc/*.{html,png,css} /usr/share/doc/expat-2.2.0
```

10.57.2. Contents of Expat

Installed program:	xmlwf
Installed library:	libexpat.[so,a]
Installed directory:	/usr/share/doc/expat-2.2.0

Short Descriptions

xmlwf is a non-validating utility to check whether or not XML documents are well formed

libexpat contains API functions for parsing XML

10.58. XML::Parser-2.44

XML::Parser is a perl module for parsing XML documents.

10.58.1. Installation of XML::Parser

Prepare XML::Parser for compilation:

```
perl Makefile.PL
```

Compile the package:

```
make
```

To test the results, issue:

```
make test
```

Install the package:

```
make install
```

10.58.2. Contents of XML::Parser

Installed program: None

Installed libraries: None

10.59. Intltool-0.51.0

The Intltool package contains internationalization tools.

10.59.1. Installation of Intltool

Apply the following patch to fix outdated syntax that is no longer recognized by Perl-5.26.0.

```
patch -Np1 -i ../intltool-0.51.0-perl-5.22-compatibility.patch
```

Prepare Intltool for compilation:

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

Compile the package:

```
make
```

To test the results, issue:

```
make check
```

Install the package:

```
make install
```

10.59.2. Contents of Intltool

Installed programs:	intltool-extract, intltool-merge, intltool-prepare, intltool-update, intltoolize
Installed directories:	/usr/share/intltool

Short Descriptions

intltool-extract	Generates header files which can be read by gettext
intltool-merge	Merges translated strings into various types of files
intltool-prepare	Prepares software to make use of intltool
intltool-update	Updates PO template files and merges translations with them
intltoolize	Copies intltool related files to software packages

10.60. Kmod-24

The Kmod package contains programs for loading, inserting and removing kernel modules for Linux. Kmod replaces the Module-Init-tools package.

10.60.1. Installation of Kmod

Prepare Kmod for compilation:

```
./configure \
  --prefix=/usr \
  --bindir=/bin \
  --sysconfdir=/etc \
  --with-rootlibdir=/lib \
  --with-zlib \
  --with-xz
```

The meaning of the configure option:

`--with-rootlibdir=/lib`
Install location for shared libraries.

`--with-zlib` `--with-xz`
This allows the Kmod package to handle zlib and XZ compressed kernel modules.

Compile the package:

```
make
```

To test the results, issue:

```
make check
```

Install the package:

```
make install
```

Create symbolic links for programs that expect Module-Init-Tools:

```
ln -sfv kmod /bin/lsmmod
for tool in depmod insmod modinfo modprobe rmmmod; do
  ln -sfv ../bin/kmod /sbin/${tool}
done
```

10.60.2. Contents of Kmod

Installed programs: depmod (link to kmod), insmod (link to kmod), kmod, lsmod (link to kmod), modinfo (link to kmod), modprobe (link to kmod), rmmmod (link to kmod)

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
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

10.61. Patch-2.7.5

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

10.61.1. 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
```

10.61.2. 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.

10.62. Psmisc-22.21

The Psmisc package contains programs for displaying information about running processes.

10.62.1. 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
```

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
```

10.62.2. Contents of Psmisc

Installed programs: fuser, killall, peekfd, prtstat, pstree, 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	Peeks at file descriptors of running processes
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

10.63. Systemd-233

The systemd package is a system and service manager for Linux operating systems.

10.63.1. Installation of Systemd

Prevent the Makefile from trying to run `setcap` on `systemd-detect-virt`, which will fail if the kernel or file system does not support extended capabilities:

```
sed -i '/virt-install-hook /d' Makefile.in
```

The `timesyncd.conf` file contains a reference to a non-existent `timesyncd.conf(5)` man page. Remove that reference to avoid possible confusion:

```
sed -i '/timesyncd.conf/d' src/timesync/timesyncd.conf.in
```

Create a file to declare some variables

```
cat > config.cache << "EOF"
KILL="/bin/kill"
MOUNT_PATH="/bin/mount"
UMOUNT_PATH="/bin/umount"
SULOGIN="/sbin/sulogin"
XSLTPROC="/usr/bin/xsltproc"
cc_cv_LDFLAGS__Wl__fuse_ld_gold=no
EOF
```

Prepare systemd for compilation:

```
./configure \
  --prefix=/usr \
  --sysconfdir=/etc \
  --localstatedir=/var \
  --libexecdir=/usr/lib \
  --docdir=/usr/share/doc/systemd-233 \
  --with-rootprefix="" \
  --with-rootlibdir=/lib \
  --enable-split-usr \
  --disable-firstboot \
  --disable-ldconfig \
  --disable-lto \
  --disable-sysusers \
  --with-default-dnssec=no \
  --with-kbd-loadkeys=/bin/loadkeys \
  --with-kbd-setfont=/bin/setfont \
  --with-dbuspolicydir=/etc/dbus-1/system.d \
  --with-dbusseessionservicedir=/usr/share/dbus-1/services \
  --with-dbusystemservedir=/usr/share/dbus-1/system-services \
  --config-cache
```

The meaning of the configure options:*--config-cache*Use the created `config.cache`.*--with-root**

These switches ensure that core programs and shared libraries are installed in the subdirectories of the root partition.

*--enable-split-usr*This switch ensures that systemd will work on systems where `/bin`, `/lib` and `/sbin` directories are not symlinks to their `/usr` counterparts.*--disable-firstboot*

This switch prevents installation of systemd services responsible for setting up the system for the first time. They are not useful for CLFS as everything is done manually.

*--disable-ldconfig*This switch prevents installation of a systemd unit that runs **ldconfig** at boot, and increases boot time. While it may be not useful for source distributions like CLFS, this option may be removed.*--disable-lto*This prevents the build system from using GCC's Link-time optimization (LTO), to ensure that systemd's binaries will not try to link to `libgcc_s`*--disable-sysusers*This switch prevents install of systemd services which setup the previously created `/etc/group` and `/etc/passwd` files.*--with-dbus**

These switches ensure that D-Bus configuration files get installed to the correct locations.

--with-default-dnssec=no

This switch turns off the experimental DNSSEC support.

Compile the package:

make

Prevent a broken test case from running:

```
sed -e 's@test/udev-test.pl @@' \
    -e 's@test-copy$(EXEEXT) @@' \
    -i Makefile.in
```

To test the results, issue:

```
sed -i "s:minix:ext4:g" src/test/test-path-util.c
make check
```

Install the package:

make install

Install documentation files that are not installed by default:

```
install -v -m644 man/*.html /usr/share/doc/systemd-233
```

Remove an unnecessary directory:

```
rm -rfv /usr/lib/rpm
```

Create symlinks for backwards-compatibility with Sysvinit:

```
for tool in runlevel reboot shutdown poweroff halt telinit; do
    ln -sfv ../bin/systemctl /sbin/$tool
done
ln -sfv ../lib/systemd/systemd /sbin/init
```

10.63.2. Configuring Systemd

Create /etc/machine-id which is needed by Journald:

```
systemd-machine-id-setup
```

Create a file to identify the operating system. **systemd** will use this file on boot to put information on the screen.

```
cat > /etc/os-release << "EOF"
# Begin /etc/os-release

NAME=Cross-LFS
ID=clfs

PRETTY_NAME=Cross Linux From Scratch
ANSI_COLOR=0;33

VERSION=GIT-20170803
VERSION_ID=20170803

# End /etc/os-release
EOF
```

10.63.3. Contents of Systemd

Installed programs:	bootctl, busctl, halt (link to systemctl), hostnamectl, init (link to systemd), journalctl, kernel-install, localectl, loginctl, machinectl, poweroff (link to systemctl), reboot (link to systemctl), runlevel (link to systemctl), shutdown (link to systemctl), systemctl, systemd, system-analyze, systemd-ask-password, systemd-cat, systemd-cgls, systemd-cgtop, systemd-coredumpctl, systemd-delta, systemd-detect-virt, systemd-inhibit, systemd-machine-id-setup, systemd-notify, systemd-nspawn, systemd-run, systemd-stdio-bridge (link to systemd-bus-proxyd), systemd-tmpfiles, systemd-tty-ask-password-agent, telinit (link to systemctl), timedatectl, udevadm
Installed libraries:	libnss_myhostname.so, libsystemd.so, libudev.so
Installed directories:	/etc/binfmt.d, /etc/init.d, /etc/kernel, /etc/modules-load.d, /etc/sysctl.d, /etc/systemd, /etc/tmpfiles.d, /etc/udev, /etc/xdg/systemd, /lib/systemd, /lib/udev, /usr/include/systemd, /usr/lib/binfmt.d, /usr/lib/kernel, /usr/lib/modules-load.d, /usr/lib/sysctl.d, /usr/lib/systemd, /usr/share/doc-systemd-233, /usr/share/systemd, /usr/share/zsh, /var/lib/systemd

Short Descriptions

bootctl	Controls the firmware and boot manager settings
busctl	Introspects and monitors the D-Bus bus
halt	Halts, powers off, or reboots the machine
hostnamectl	Controls the system hostname
init	systemd system and service manager
journalctl	Queries the systemd journal
kernel-install	Adds and removes kernel and initramfs images to and from <code>/boot</code>
localectl	Controls the system locale and keyboard layout settings
loginctl	Controls the systemd login manager
machinectl	Controls the systemd machine manager
poweroff	Halts, powers off, or reboots the machine
reboot	Halts, powers off, or reboots the machine
runlevel	Prints previous and current SysV runlevel
shutdown	Halts, powers off, or reboots the machine
systemctl	Control the systemd system and service manager
systemd	System and service manager for Linux
systemd-analyze	Analyzes system boot-up performance
systemd-ask-password	Queries the user for a system passphrase, via the TTY or an UI agent.
systemd-cat	Connects a pipeline or program's output with the journal
systemd-cgls	Recursively shows control group contents
systemd-cgtop	Shows top control groups by resource usage
systemd-coredumpctl	Retrieves coredumps from the journal
systemd-delta	Finds overridden configuration files
systemd-detect-virt	Detects execution in a virtual environment
systemd-inhibit	Executes a program with an inhibition lock taken
systemd-machine-id-setup	Initializes the machine ID in <code>/etc/machine-id</code>
systemd-notify	Notifies init system about start-up completion and other daemon status changes
systemd-nspawn	Spawns a namespace container for debugging, testing, and building
systemd-run	Runs programs in transient scope or service units
systemd-stdio-bridge	Connects stdio or a socket to a given bus address
systemd-tmpfiles	Creates, deletes, and cleans up volatile and temporary files
systemd-tty-ask-password-agent	Process system password requests
telinit	Tells init which run-level to change to
timedatectl	Controls the system time and date

udevadm

Udev management tool

libnss_myhostname

Plugin for the GNU Name Service Switch (NSS) functionality of Glibc, providing hostname resolution for the locally configured system hostname

libsystemd

Support library for systemd

libudev

A library interface to Udev device information.

10.64. D-Bus-1.10.18

D-Bus is a message bus system, a simple way for applications to talk to one another.

10.64.1. Installation of D-Bus

Prepare D-Bus for compilation:

```
./configure \
  --prefix=/usr \
  --sysconfdir=/etc \
  --libexecdir=/usr/lib/dbus-1.0 \
  --localstatedir=/var \
  --with-systemdsystemunitdir=/lib/systemd/system \
  --docdir=/usr/share/doc/dbus-1.10.18
```

Compile the package:

```
make
```

This package does come with a test suite, but it requires several packages that are not included in CLFS. Instructions for running the test suite can be found in the CBLFS wiki at http://cblfs.clfs.org/index.php/D-BUS_Core.

Install the package:

```
make install
```

Move the shared library to `/lib` and recreate the symbolic link.

```
mv -v /usr/lib/libdbus-1.so.* /lib
ln -sfv ../../lib/${readlink /usr/lib/libdbus-1.so} /usr/lib/libdbus-1.so
```

Create a symlink so that D-Bus and systemd can use the same `machine-id` file:

```
ln -sv /etc/machine-id /var/lib/dbus
```

10.64.2. Contents of D-Bus

Installed programs:	dbus-cleanup-sockets, dbus-daemon, dbus-launch, dbus-monitor, dbus-send, dbus-uuidgen
Installed libraries:	libdbus-1.[a,so]
Installed directories:	/etc/dbus-1, /usr/include/dbus-1.0, /usr/lib/dbus-1.0, /usr/share/doc/dbus-1.10.18, /var/lib/dbus

Short Descriptions

dbus-cleanup-sockets	Cleans up leftover sockets in a directory
dbus-daemon	The message bus daemon
dbus-uuidgen	Utility to generate UUIDs
dbus-monitor	A debug probe that prints message bus messages
dbus-launch	Utility to start a message bus from a shell script

dbus-send

Send a message to a message bus

libdbus-1

Library containing the API for using the message bus

10.65. Tar-1.29

The Tar package contains an archiving program.

10.65.1. Installation of Tar

Prepare Tar for compilation:

```
FORCE_UNSAFE_CONFIGURE=1 ./configure \  
  --prefix=/usr \  
  --bindir=/bin \  
  --libexecdir=/usr/sbin
```

Compile the package:

```
make
```

To test the results, issue:

```
make check
```

Install the package:

```
make install
```

Install the documentation:

```
make -C doc install-html docdir=/usr/share/doc/tar-1.29
```

10.65.2. Contents of Tar

Installed programs:	rmt, tar
Installed directory:	/usr/share/doc/tar-1.29

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

10.66. Texinfo-6.3

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

10.66.1. 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
```

If TeX will be used, install the components belonging in a TeX installation:

```
make TEXMF=/usr/share/texmf install-tex
```

10.66.2. Contents of Texinfo

Installed programs:	info, infokey, install-info, makeinfo (link to texi2any), pdftexi2dvi, texi2dvi, texi2pdf, 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 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	Shell script that run texi2dvi --pdf
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

10.67. Util-linux-2.29.2

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

10.67.1. 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:

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

10.67.2. Installation of Util-linux

Prepare Util-linux for compilation:

```
./configure \
  ADJTIME_PATH=/var/lib/hwclock/adjtime \
  --enable-write \
  --disable-chfn-chsh \
  --disable-login \
  --disable-nologin \
  --disable-su \
  --disable-setpriv \
  --disable-runuser \
  --docdir=/usr/share/doc/util-linux-2.29.2
```

The meaning of the configure options:

--enable-write

This option allows the **write** program to be installed.

*--disable-**

This option disables various programs

Compile the package:

```
make
```

To test the results, issue:

```
chown -Rv nobody . &&
su nobody -s /bin/bash -c "PATH=$PATH make -k check"
```

Install the package:

```
make install
```

10.67.3. Contents of Util-linux

Installed programs:	addpart,agetty,blkdiscard,blkid,blockdev,cal,cfdisk,chcpu,chrt,col,colcrt,colrm,column,ctrlaltdel,delpart,dmesg,eject,fallocate,fdformat,fdisk,findfs,findmnt,flock,fsck,fsck.cramfs,fsck.minix,fsfreeze,fstrim,getopt,hexdump,hwclock,ionice,ipcmk,ipcrm,ipcs,isosize,kill,last,lastb (link to last),ldattach,logger,look,losetup,lsblk,lscpu,lslocks,mcookie,mesg,mkfs,mkfs.bfs,mkfs.cramfs,mkfs.minix,mkswap,more,mount,mountpoint,namei,nsenter,partx,pg,pivot_root,prlimit,raw,readprofile,rename,renice,resizepart,rev,rtcwake,script,scriptreplay,setarch,setsid,setterm,sfdisk,sulogin,swapon,swapon,switch_root,tailf,taskset,ul,umount,unshare,utmpdump,uudd,uuidgen,wall,wdctl,whereis,wipefs,write
Installed libraries:	libblkid.[a,so], libmount.[a,so], libuuid.[a,so]
Installed directories:	/usr/include/blkid, /usr/include/libmount, /usr/include/uuid, /usr/share/bash-completion, /usr/share/doc/util-linux-2.29.2/getopt, and /var/lib/hwclock

Short Descriptions

addpart	Notifies the kernel of a new partition
agetty	Opens a tty port, prompts for a login name, and then invokes the login program
blkdiscard	Discards sectors on a device
blkid	A command line utility to locate and print block device attributes
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	Utility to configure 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
delpart	Asks the kernel to remove a partition
dmesg	Dumps the kernel boot messages
eject	Eject removable media
fallocate	Preallocates space to a file
fdformat	Low-level formats a floppy disk
fdisk	Manipulates the partition table of the given device
findfs	Finds a file system by label or Universally Unique Identifier (UUID)
findmnt	Lists mounted filesystems or searches for a filesystem
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	Suspends and resumes access to a filesystem
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
ionice	Gives and sets program I/O scheduling class and priority
ipcmk	Creates various IPC resources
ipcrm	Removes the given Inter-Process Communication (IPC) resource
ipcs	Provides IPC status information
isozsize	Reports the size of an iso9660 file system
kill	Send a signal to a process
last	Shows which users last logged in (and out), searching back through the <code>/var/log/wtmp</code> file; it also shows system boots, shutdowns, and run-level changes
lastb	Shows the failed login attempts, as logged in <code>/var/log/btmp</code>
ldattach	Attaches a line discipline to a serial line
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	Prints information about block devices
lscpu	Prints CPU architecture information
lslocks	Lists local system locks
mcookie	Generates magic cookies (128-bit random hexadecimal numbers) for xauth
mesg	Controls whether other users can send messages to the current user's terminal
mkfs	Builds a file system on a device (usually a hard disk partition)
mkfs.bfs	Creates a Santa Cruz Operations (SCO) bfs file system
mkfs.cramfs	Creates a cramfs file system
mkfs.minix	Creates a Minix file system
mkswap	Initializes the given device or file to be used as a swap area
more	A filter for paging through text one screen at a time
mount	Attaches the file system on the given device to a specified directory in the file-system tree
mountpoint	Tells you whether or not a directory is a mount point.
namei	Shows the symbolic links in the given pathnames
nsenter	Runs a program with namespaces of other processes

partx	Tells the kernel about the presence and numbering of on-disk partitions
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	Gets and sets a process' resource limits
raw	Binds 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	Enters a system sleep state until a specified wakeup time
script	Makes a typescript of a terminal session
scriptreplay	Plays back typescripts created by script
setarch	Changes reported architecture in new program environment and sets personality flags
setsid	Runs the given program in a new session
setterm	Sets terminal attributes
sfdisk	A disk partition table manipulator
sulogin	Allows <i>root</i> to log in; it is normally invoked by init when the system goes into single user mode
swapon	Enables devices and files for paging and swapping and lists the devices and files currently in use
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's CPU affinity
ul	A filter for translating underscores into escape sequences indicating underlining for the terminal in use
umount	Disconnects a file system from the system's file tree
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
uudd	A daemon used by the UUID library to generate time-based UUIDs in a secure and guaranteed-unique fashion.
uuddgen	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	Writes a message to all logged-in users
wdctl	Show 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
write	Sends a message to the given user <i>if</i> that user has not disabled receipt of such messages
libblkid	Contains routines for device identification and token extraction
libmount	Contains routines for parsing the <code>/etc/fstab</code> , <code>/etc/mtab</code> , and <code>/proc/self/mountinfo</code> files, managing <code>/etc/mtab</code> , and configuring various mount options
libuuid	Contains routines for generating unique identifiers for objects that may be accessible beyond the local system

10.68. Vim-8.0

The Vim package contains a powerful text editor.

10.68.1. Installation of Vim

Alternatives to Vim

If you prefer another editor—such as Emacs, Joe, or Nano—please refer to http://cblfs.clfs.org/index.php/Category:Text_Editors for suggested installation instructions.

The following patch merges all updates from the 8.0 Branch from the Vim developers:

```
patch -Np1 -i ../vim-8.0-branch_update-1.patch
```

Change the default location of the `vimrc` configuration file to `/etc`:

```
echo '#define SYS_VIMRC_FILE "/etc/vimrc"' >> src/feature.h
```

Prepare Vim for compilation:

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

Compile the package:

```
make
```

To test the results, issue:

```
make test
```

However, this test suite outputs a lot of binary data to the screen, which can cause issues with the settings of the current terminal. This can be resolved by redirecting the output to a log file.

Install the package:

```
make -j1 install
```

Many users are accustomed to using **vi** instead of **vim**. Some programs, such as **vigr** and **vipw**, also use **vi**. Create a symlink to permit execution of **vim** when users habitually enter **vi** and allow programs that use **vi** to work:

```
ln -sv vim /usr/bin/vi
```

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-8.0`, making it consistent with the location of documentation for other packages:

```
ln -sv ../vim/vim0597/doc /usr/share/doc/vim-8.0
```

If an X Window System is going to be installed on the CLFS system, you may want 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, refer to the Vim documentation and the Vim installation page in CBLFS at <http://cblfs.clfs.org/index.php/Vim>.

10.68.2. 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
set ruler
syntax on
if (&term == "item") || (&term == "putty")
    set background=dark
endif

" End /etc/vimrc
EOF
```

The *set nocompatible* 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* allows backspacing over line breaks, autoindents, and the start of insert. The *syntax on* enables vim's syntax highlighting. Finally, the *if* statement with the *set background=dark* 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'
```

10.68.3. Contents of Vim

Installed programs:	efm_filter.pl, efm_perl.pl, ex (link to vim), less.sh, mve.awk, pltags.pl, ref, rview (link to vim), rvim (link to vim), shtags.pl, tcltags, vi (link to vim), view (link to vim), vim, viml32, vim2html.pl, vimdiff (link to vim), vimmm, vimspell.sh, vimtutor, xxd
Installed directory:	/usr/share/vim

Short Descriptions

efm_filter.pl	A filter for creating an error file that can be read by vim
efm_perl.pl	Reformats the error messages of the Perl interpreter for use with the “quickfix” mode of vim
ex	Starts vim in ex mode
less.sh	A script that starts vim with less.vim
mve.awk	Processes vim errors
pltags.pl	Creates a tags file for Perl code for use by vim
ref	Checks the spelling of arguments

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
shtags.pl	Generates a tags file for Perl scripts
tcltags	Generates a tags file for TCL code
view	Starts vim in read-only mode
vi	Link to vim
vim	Is the editor
vim132	Starts vim with the terminal in 132-column mode
vim2html.pl	Converts Vim documentation to HypterText Markup Language (HTML)
vimdiff	Edits two or three versions of a file with vim and show differences
vimm	Enables the DEC locator input model on a remote terminal
vimspell.sh	Spell checks a file and generates the syntax statements necessary to highlight in vim . This script requires the old Unix spell command, which is provided neither in CLFS nor in CBLFS
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

10.69. Colo-1.22

The Colo package contains the Cobalt Boot Loader.

10.69.1. Installation of Colo

Note

This bootloader is for the MIPS based cobalt servers RaQ, RaQ2, Qube, or the Qube2.

This patch fixes a relocation error when linking with Binutils:

```
patch -Np1 -i ../colo-1.22-relocation_fix-1.patch
```

Disable -Werror in one of the Makefiles to prevent a compile failure:

```
sed -i '/^CFLAGS/s/-Werror //' stage2/Makefile
```

Compile the Colo package:

```
make
```

Install the package:

```
install -dv /usr/lib/colo/examples
install -v chain/colo-chain.elf /usr/lib/colo
install -v tools/lcdtools/e2fsck-lcd/e2fsck-lcd /sbin
install -v tools/lcdtools/e2fsck-lcd/e2fsck-lcd.8 /usr/man/man8
install -v tools/lcdtools/paneld/paneld /sbin
install -v tools/lcdtools/paneld/paneld.8 /usr/man/man8
install -v tools/lcdtools/putlcd/putlcd /sbin
install -v tools/lcdtools/putlcd/putlcd.8 /usr/man/man8
install -v examples/menu.colo /usr/lib/colo/examples
install -v examples/simple.colo /usr/lib/colo/examples
cp -v chain/colo-chain.elf /boot/vmlinux
gzip -9 /boot/vmlinux
```

10.69.2. Contents of Colo

Installed programs: colo-chain.elf, e2fsck-lcd, paneld and putlcd

Short Descriptions

colo	Is the Cobalt Bootloader's chain mode executeable. This file gets gzipped and renamed to vmlinux.gz, so it can be booted automatically by the Cobalt's existing firmware
e2fsck-lcd	Will output file system check progress information on the Cobalt LCD
paneld	Is an admin tool for the LCD panel of Cobalt machines. By default, it will display the current time and optionally a message. When you hold the enter or select button for a couple of seconds you will get an admin menu. The menu will allow you to either halt or reboot your Cobalt machine
putlcd	Is a tool to display text on the LCD display of Cobalt machines

md5rom

Will output the MD5 checksum of a Cobalt's ROM

10.70. Dvhtool-1.0.1

The Dvhtool package is used to manipulate volume headers of devices using sgi disk labels.

10.70.1. Installation of Dvhtool

Note

This program is required for the Arcload bootloader, which is for SGI Workstations and SGI Servers based on MIPS Processors.

This patch fixes build issues with Dvhtool and adds support for LVM and Linux partitions:

```
patch -Np1 -i ../dvhtool-1.0.1-fixes-1.patch
```

Prepare Dvhtool for compilation:

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

Compile the package:

```
make
```

Install the package:

```
make install
```

10.70.2. Contents of Dvhtool

Installed programs: dvhtool

Short Descriptions

dvhtool A utility for displaying SGI disk partition and volume header information as well as for copying files to and from the volume header

10.71. Arcload-0.5

The Arcload package contains a SGI Bootloader.

10.71.1. Installation of Arcload

Note

This program is the Arcload bootloader, which is for SGI Workstations and SGI Servers based on MIPS Processors.

Compile the package:

```
make MODE=M32 clean
make CC="gcc" LD="ld" MODE=M32
```

Install the package:

```
install -dv /usr/lib/arcload
cp -v arcload.ecoff /usr/lib/arcload/sash
```

10.71.2. Contents of Arcload

Installed programs: sashARCS or sash64

Short Descriptions

sashARCS Is a bootloader for IP22/IP32 MIPS systems, which are 32 Bit.

sash64 Is a bootloader for IP27/IP28/IP30 MIPS systems, which are 64 Bit.

10.72. 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.

10.73. Stripping

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 200 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 current situation.

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 If You Are Going to Chroot first exit from chroot:

```
logout
```

Then reenter it with:

```
chroot ${CLFS} /tools/bin/env -i \
    HOME=/root TERM=${TERM} PS1='\u:\w:$ ' \
    PATH=/bin:/usr/bin:/sbin:/usr/sbin \
    /tools/bin/bash --login
```

Now the binaries and libraries can be safely stripped:

```
/tools/bin/find /{,usr/}{bin,lib,sbin} -type f \
    -exec /tools/bin/strip --strip-debug '{}' ';'
```

A large number of files will be reported as having their file format not recognized. These warnings can be safely ignored. These warnings indicate that those files are scripts instead of binaries.

If disk space is very tight, the `--strip-all` option can be used on the binaries in `/ { ,usr/ } {bin,sbin}` to gain several more megabytes. Do not use this option on libraries—they will be destroyed.

Chapter 11. System Configuration

11.1. Introduction

This chapter details how to finish configuring the base system. This includes some final configuration for systemd as well as locales and a simple bash profile that should be suitable for most users.

11.2. How does Systemd work?

Warning

Please disregard this page until it is complete and verified.

11.2.1. Introduction to Systemd

Systemd is a system management daemon designed exclusively for the Linux kernel API. In the Linux startup process, it is the first process to execute in user land; therefore, it is also the parent process of all child processes in user land.

Systemd's initialization instructions for each daemon are recorded in a declarative configuration file rather than a shell script. For inter-process communication, systemd makes Unix domain sockets and D-Bus available to the running daemons. Because systemd tracks processes using Linux cgroups instead of process identifiers (PIDs), daemons cannot "escape" systemd; not even by double-forking. Systemd is also capable of aggressive parallelization.

Among systemd's auxiliary features are a cron-like job scheduler called systemd Calendar Timers, and an event logging subsystem called journal. The system administrator may choose whether to log system events with systemd or syslog. Systemd's logfile is a binary file. The state of systemd itself can be preserved in a snapshot for future recall.

Systemd provides a replacement for sysvinit, pm-utils, inetd, acpid, syslog, watchdog, cron and atd, and obsoletes ConsoleKit.

11.2.2. Systemctl

systemctl is the main command used to introspect and control systemd.

List running units:

systemctl or **systemctl list-units**

List failed units:

systemctl --failed

List available unit files:

systemctl list-unit-files

Activate a unit immediately:

systemctl start *unit*

Stop a unit immediately:

systemctl stop *unit*

Restart a unit:

systemctl restart *unit*

Reload unit configuration:

```
systemctl reload unit
```

Show status of a unit:

```
systemctl status unit
```

Check if a unit is enabled or disabled:

```
systemctl is-enabled unit
```

Enable a unit to start during boot:

```
systemctl enable unit
```

Disable a unit to not start during boot:

```
systemctl disable unit
```

Reload systemd and scan for new or changed units:

```
systemctl daemon-reload
```

For more information regarding systemd, please refer to the systemd and related man-pages and *Systemd at FedoraProject* for documentation, examples, features, and other information.

11.3. Configuring the system clock

This section discusses how to configure the **systemd-timedated** system service, which configures system clock and timezone.

Systemd provides a **timedatectl** utility which is used to communicate with **systemd-timedated**. It can be used to set the system clock in local time or UTC time, depending on the hardware clock setting. By default, **systemd-timedated** will assume that clock is set to UTC time.

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.

systemd-timedated reads `/etc/adjtime`, and depending on the contents of the file, it sets the clock to either UTC or local time.

Create the `/etc/adjtime` file with the following contents if your hardware clock is set to local time:

```
cat > /etc/adjtime << "EOF"
0.0 0 0.0
0
LOCAL
EOF
```

If `/etc/adjtime` isn't present at first boot, **systemd-timedated** will assume that hardware clock is set to UTC and adjust the file according to that.

If your clock is set to local time, tell **systemd-timedated** about it by running the following command:

```
timedatectl set-local-rtc 1
```

timedatectl can also be used to change system time and time zone.

To change your current system time, issue:

```
timedatectl set-time YYYY:MM:DD HH:MM:SS
```

Hardware clock will also be updated accordingly.

To change your current time zone, issue:

```
timedatectl set-timezone TIMEZONE
```

You can get list of available time zones by running:

```
timedatectl list-timezones
```

Note

Please note that **timedatectl** command can be used only on a system booted with systemd.

11.4. Configuring the Linux Console

This section discusses how to configure the **systemd-vconsole-setup** system service which configures the virtual console font and console keymap. The **systemd-vconsole-setup** service reads `/etc/vconsole.conf` for configuration information. Decide which keymap and screen font will be used. Various language-specific HOWTO's can help. with this (see <http://www.tldp.org/HOWTO/HOWTO-INDEX/other-lang.html>). Examine **localectl list-keymaps** output for a list of valid console keymaps. Look in `/usr/share/consolefonts` for valid screen fonts.

The `/etc/vconsole.conf` file should contain lines of the form: `VARIABLE="value"`. The following variables are recognized:

KEYMAP

This variable specifies the key mapping table for the keyboard. If unset, it defaults to `us`.

KEYMAP_TOGGLE

This variable can be used to configure a second toggle keymap and is unset by default.

FONT

This variable specifies the font used by the virtual console.

FONT_MAP

This variable specifies the console map to be used.

FONT_UNIMAP

This variable specifies the unicode font map.

An example for a German keyboard and console is given below:

```
cat > /etc/vconsole.conf << "EOF"
KEYMAP=de-latin1
FONT=Lat2-Terminus16
EOF
```

You can change KEYMAP value at runtime by using the **localectl** utility:

```
localectl set-keymap MAP
```

Note

Please note that **localectl** command can be used only on a system booted with systemd.

You can also use **localectl** utility with the corresponding parameters to change X11 keyboard layout, model, variant and options:

```
localectl set-x11-keymap LAYOUT [MODEL] [VARIANT] [OPTIONS]
```

To list possible values for **localectl set-x11-keymap** parameters, run **localectl** with parameters listed below:

list-x11-keymap-models

Show known X11 keyboard mapping models.

list-x11-keymap-layouts

Show known X11 keyboard mapping layouts.

list-x11-keymap-variants

Show known X11 keyboard mapping variants.

list-x11-keymap-options

Show known X11 keyboard mapping options.

Note

Using any of the parameters listed above requires *XKeyboard Client* package from CBLFS.

11.5. Device and Module Handling on a CLFS System

In Installing Basic System Software, we installed Udev, as one of the components of systemd. Before we go into the details regarding how this works, a brief history of previous methods of handling devices is in order.

11.5.1. History

11.5.1.1. Static Device Nodes

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.

11.5.1.2. Devfs

In February 2000, a new filesystem called `devfs`, which dynamically created device nodes as devices were found by the kernel, 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 suffered from race conditions that were inherent in its design and could not be fixed without a substantial revision to the kernel. It was marked deprecated with the release of the 2.6 kernel series, and was removed entirely as of version 2.6.18.

11.5.1.3. Sysfs

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. Drivers that have been compiled into the kernel directly register their objects with `sysfs` 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 built-in drivers registered with `sysfs` are available to userspace processes. With this userspace-visible representation, the possibility of seeing a userspace replacement for `devfs` became much more realistic.

11.5.1.4. Udev Implementation

Shortly after the introduction of `sysfs`, work began on a program called Udev to advantage of it. The **udev** daemon made calls to `mknod()` to create device nodes in `/dev` dynamically, based on the information from `sysfs`, in `/sys`. For example, `/sys/class/tty/vcs/dev` contains the string “7:0”. This string was used by **udev** to create a device node with major number 7 and minor number 0.

Linux kernel version 2.6.32 introduced a new virtual file system called `devtmpfs`, an improved replacement for `devfs`. This allows device nodes to once again be dynamically created by the kernel, without many of the problems of `devfs`. As of version 176, Udev no longer creates device nodes itself, instead relying on `devtmpfs` to do so.

11.5.1.5. Systemd and Eudev

In 2010, development began on `systemd`, an alternate **init** implementation. Starting with Udev 183, Udev's source tree was merged with `systemd`. Several Gentoo developers who disagreed with this merge announced a project fork called Eudev in December 2012, created by extracting the Udev code from `systemd`. One of the goals of Eudev is to allow for easier installation and usage of **udev** without the need for the rest of `systemd`.

11.5.2. Device Node Creation

By default, device nodes created by the kernel in a `devtmpfs` are owned by `root:root` and have `600` permissions. **udev** can modify ownership and permissions of the nodes under the `/dev` directory, and can also create additional symlinks, based on rules specified in the files within the `/etc/udev/rules.d`, `/lib/udev/rules.d`, and `/run/udev/rules.d` directories. The names for these files start with a number, to indicate the order in which they are run, and they have a `.rules` extension (**udev** will ignore files with any other extension). All of the rules files from these directories are combined into a single list, sorted by filename, and run in that order. In the event of a conflict, where a rules file with the same name exists in two or more of these directories, the rules in `/etc` take the highest priority, followed by rules files in `/run`, and finally `/lib`. Any device for which a rule cannot be found will just be ignored by **udev** and be left at the defaults defined by the kernel, as described above. For more details about writing Udev rules, see </usr/share/doc/systemd-233/udev.html>.

11.5.3. 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 by Udev will cause **udev** to call out to `/sbin/modprobe` with the contents of the `MODALIAS` uevent environment variable (that 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.

11.5.4. Problems with Loading Modules and Creating Devices

There are a few possible problems when it comes to automatically creating device nodes.

11.5.4.1. 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-4.9.21, 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-4.9.21, 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.

11.5.4.2. 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 an “install” line to a file in `/etc/modprobe.d`. For example:

```
install snd-pcm /sbin/modprobe -i snd-pcm ; \
    /sbin/modprobe snd-pcm-oss ; true
```

If the module in question is not a wrapper and is useful by itself, configure the **S05modules** bootscript to load this module on system boot. To do this, add the module name to the `/etc/sysconfig/modules` file on a separate line. This works for wrapper modules too, but is suboptimal in that case.

11.5.4.3. Udev loads some unwanted module

Either don't build the module, or blacklist it in `/etc/modprobe.d` 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.

11.5.4.4. Udev makes a wrong symlink

This usually happens if a rule unexpectedly matches a device. For example, a poorly-written 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 **udevadm info**.

11.5.4.5. 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. Please notify the CLFS Development list if you do so and it helps.

11.5.4.6. 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 11.6, “Creating custom symlinks to devices” and Networking Configuration for examples.

11.5.5. 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>

11.6. Creating custom symlinks to devices

11.6.1. 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. Also, it may be convenient to put references to those symlinks into `/etc/fstab`. For each of your CD-ROM devices, 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.

There are two approaches to creating symlinks. The first one is to use the model name and the serial number, the second one is based on the location of the device on the bus. If you are going to use the first approach, create a file similar to the following:

```
cat >/etc/udev/rules.d/82-cdrom.rules << EOF

# Custom CD-ROM symlinks
SUBSYSTEM=="block", ENV{ID_MODEL}=="SAMSUNG_CD-ROM_SC-148F", \
    ENV{ID_REVISION}=="PS05", SYMLINK+="cdrom"
SUBSYSTEM=="block", ENV{ID_MODEL}=="PHILIPS_CDD5301", \
    ENV{ID_SERIAL}=="5VO1306DM00190", SYMLINK+="cdrom1 dvd"

EOF
```

Note

Although the examples in this book work properly, be aware that Udev does not recognize the backslash for line continuation. If modifying Udev rules with an editor, be sure to leave each rule on one physical line.

This way, the symlinks will stay correct even if you move the drives to different positions on the IDE bus, but the `/dev/cdrom` symlink won't be created if you replace the old SAMSUNG CD-ROM with a new drive.

The `SUBSYSTEM=="block"` key is needed in order to avoid matching SCSI generic devices. Without it, in the case with SCSI CD-ROMs, the symlinks will sometimes point to the correct `/dev/srX` devices, and sometimes to `/dev/sgX`, which is wrong.

The second approach yields:

```
cat >/etc/udev/rules.d/82-cdrom.rules << EOF

# Custom CD-ROM symlinks
SUBSYSTEM=="block", ENV{ID_TYPE}=="cd", \
    ENV{ID_PATH}=="pci-0000:00:07.1-ide-0:1", SYMLINK+="cdrom"
SUBSYSTEM=="block", ENV{ID_TYPE}=="cd", \
    ENV{ID_PATH}=="pci-0000:00:07.1-ide-1:1", SYMLINK+="cdrom1 dvd"

EOF
```

This way, the symlinks will stay correct even if you replace drives with different models, but place them to the old positions on the IDE bus. The `ENV{ID_TYPE}=="cd"` key makes sure that the symlink disappears if you put something other than a CD-ROM in that position on the bus.

Of course, it is possible to mix the two approaches.

11.6.2. Dealing with duplicate devices

As explained in Section 11.5, “Device and Module Handling on a CLFS 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 Networking Configuration, and sound card configuration can be found in *CBLFS*.

For each of your devices that is likely to have this problem (even if the problem doesn't exist in your current Linux distribution), find the corresponding directory under `/sys/class` or `/sys/block`. For video devices, this may be `/sys/class/video4linux/videoX`. Figure out the attributes that identify the device uniquely (usually, vendor and product IDs and/or serial numbers work):

```
udevadm info -a -p /sys/class/video4linux/video0
```


Then write rules that create the symlinks, e.g.:

```
cat >/etc/udev/rules.d/83-duplicate_devs.rules << EOF

# Persistent symlinks for webcam and tuner
KERNEL=="video*", SYSFS{idProduct}=="1910", SYSFS{idVendor}=="0d81", \
    SYMLINK+="webcam"
KERNEL=="video*", SYSFS{device}=="0x036f", SYSFS{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.

11.7. 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, and *Bash Startup Files* in CBLFS.

The files `/etc/profile` and `~/.bash_profile` are read when the shell is invoked as an interactive login shell. Create a base `/etc/profile` that will read locale information from `/etc/locale.conf` and load any Bash auto completion files that may be on the system. This script also sets the `INPUTRC` environment variable that makes Bash and Readline use `/etc/inputrc`:

```
cat > /etc/profile << "EOF"
# Begin /etc/profile

source /etc/locale.conf

for f in /etc/bash_completion.d/*
do
    if [ -e ${f} ]; then source ${f}; fi
done
unset f

export INPUTRC=/etc/inputrc

# End /etc/profile
EOF
```

11.8. Setting Up Locale Information

The `/etc/locale.conf` 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” or “US”). `[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
```

Locales can have a number of synonyms, e.g. “ISO-8859-1” is also referred to as “iso8859-1” and “iso88591”. Some applications cannot handle the various synonyms correctly, so it is safest 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_US.utf8” in our example).

```
LC_ALL=[locale name] locale charmap
```

For the “en_US.utf8” locale, the above command will print:

```
UTF-8
```

This results in a final locale setting of “en_US.UTF-8”. It is important that the locale found using the heuristic above is tested prior to it being added to `/etc/locale.conf`:

```
LC_ALL=[locale name] locale territory
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 10 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 CLFS 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:

```
Warning: locale not supported by Xlib, locale set to C
```

Sometimes it is possible to fix this by removing the charmap part of the locale specification, as long as that does not change the character map that Glibc associates with the locale (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/locale.conf` file:

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

LANG=[ll]_[CC].[charmap][@modifiers]

# End /etc/locale.conf
EOF
```

Note that you can modify `/etc/locale.conf` with systemd's **localectl** utility. To use **localectl** for the example above, run:

```
localectl set-locale LANG="[ll]_[CC][charmap][@modifiers]"
```

You can also specify other language specific environment variables such as `LANG`, `LC_CTYPE`, `LC_NUMERIC` or any other environment variable from **locale** output. Just separate them with a space. An example where `LANG` is set as `en_US.UTF-8` but `LC_CTYPE` is set as just `en_US` is:

```
localectl set-locale LANG="en_US.UTF-8" LC_CTYPE="en_US"
```

Note

Please note that **localectl** command can be used only on a system booted with systemd.

Setting the keyboard layout, screen font, and locale-related environment variables are the only internationalization steps needed to support locales that use ordinary single-byte encodings and left-to-right writing direction. UTF-8 has been tested on the English, French, German, Italian, and Spanish locales. All other locales are untested. If you discover issues with any other locale please open a ticket in our Trac system.

Some locales need additional programs and support. CLFS will not be supporting these locales in the book. We welcome the support for these other locales via <http://cblfs.clfs.org/>.

11.9. Creating the `/etc/inputrc` File

The `/etc/inputrc` file deals with mapping the keyboard for specific situations. This file is the start-up 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 inside 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
```

11.10. 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

# End /etc/fstab
EOF
```

Replace `[xxx]`, `[yyy]`, and `[fff]` with the values appropriate for the system, for example, `sda2`, `sda5`, and `ext2`. For details on the six fields in this file, see **man 5 fstab**.

Chapter 12. Networking Configuration

12.1. Configuring the system hostname

Systemd reads `/etc/hostname` to determine which hostname should be set.

Create the `/etc/hostname` file and enter a hostname by running:

```
echo "[clfs]" > /etc/hostname
```

`[clfs]` needs to be replaced with the name given to the computer. Do not enter the Fully Qualified Domain Name (FQDN) here. That information will be put in the `/etc/hosts` file in the next section.

12.2. 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 IP address could be 192.168.1.1. A valid FQDN for this IP could be `www.clfs.org` (not recommended because this is a valid registered domain address and could cause domain name server issues).

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
::1      localhost
[192.168.1.1] [<HOSTNAME>.example.org] [HOSTNAME] [alias ...]

# End /etc/hosts (network card version)
EOF
```

The `[192.168.1.1]` and `[<HOSTNAME>.example.org]` values need to be changed for specific users 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
::1      localhost

# End /etc/hosts (no network card version)
EOF
```

The `::1` entry is the IPv6 counterpart of `127.0.0.1` and represents the IPv6 loopback interface.

12.3. 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`. If at least one of your network interfaces is going to be configured by DHCP then you may not need to create this file. By default DHCPd will overwrite this file when it gets a new lease from the DHCP server. If you wish to manually configure your network interfaces or manually set your DNS using DHCP then 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.

12.4. Systemd Networking?

This section only applies if a network card is to be configured. If you do not need to configure a network interface you can skip on to Making the CLFS System Bootable.

There are two different ways you can proceed from this point to configure your network. You can use `systemd`, or install the CLFS-Network-Scripts.

To use `systemd` to configure a Network Interface, Follow Section 12.5, “Networking Configuration with Systemd-networkd”.

To use CLFS-network-scripts to configure a Network Interface, Follow Section 12.6, “CLFS-Network-Scripts-20140224”.

12.5. Networking Configuration with Systemd-networkd

12.5.1. Network Interface Configuration

Note

Udev may assign random Network Card Interface names for some network cards such as enp2s1. If you are not sure what your Network Card Interface name is, you can always run **ip l** after you have booted your system. It is important that the Name variable in `/etc/systemd/network` contain the correct Network Card Interface name (e.g. Name=enp2s1 or Name=eth0) or systemd will fail to bring up your network interface.

12.5.1.1. Static Network Interface Configuration

systemd-networkd uses `/etc/systemd/network` for configuration files. Refer to `systemd.network(5)` and `systemd.netdev(5)`. Configure a network interface with a config file. Adjust Name= as required:

```
cd /etc/systemd/network &&
cat > static.network << "EOF"
[Match]
Name=enp2s0

[Network]
Address=192.168.1.1/24
Gateway=192.168.1.2
EOF
```

The values of these variables must be changed in every file to match the proper setup.

The Name variable defines the interface name, for example, eth0. It is required for all network device configuration files.

The Gateway variable should contain the default gateway IP address, if one is present. If not, then comment out the variable entirely.

For more information see the **systemd.netdev** man page.

12.5.1.2. Connecting to a network with DHCP

systemd-networkd uses `/etc/systemd/network` for configuration files. Refer to `systemd.network(5)` and `systemd.netdev(5)`. Configure a network interface with a config file. Adjust Name= as required:

```
cd /etc/systemd/network &&
cat > dhcp.network << "EOF"
[Match]
Name=enp2s0

[Network]
DHCP=yes
EOF
```


systemd-networkd will automatically configure `/run/systemd/network/resolv.conf` when using DHCP. If you did not manually create `/etc/resolv.conf`, create a symlink:

```
ln -sv /run/systemd/network/resolv.conf /etc
```

12.5.2. Using Timesyncd

Systemd includes a simple NTP client daemon, **systemd-timesyncd**, though it is disabled by default. If you want to enable it, you will first need to add a required user and group:

```
groupadd -g 78 systemd-timesync  
useradd -g systemd-timesync -u 78 -d /dev/null -s /bin/false systemd-timesync
```

Then, actually enable **systemd-timesyncd** so that it will run on system boot:

```
systemctl enable systemd-timesyncd
```

You can configure **systemd-timesyncd** by editing `/etc/systemd/timesyncd.conf`.

Continue to Making the CLFS System Bootable.

12.6. CLFS-Network-Scripts-20140224

The CLFS-Network-Scripts package contains a set of scripts to configure the network at bootup and deconfigure it at shutdown.

12.6.1. Installation of CLFS-Network-Scripts

Install the package:

```
make install
```

12.6.2. Contents of CLFS-Network-Scripts

Installed scripts:	ifdown, ifup, ipv4-static
Installed systemd units:	ifupdown@.service, dhcpcd@.service, nscd.service
Installed directories:	/etc/sysconfig, /lib/services, /lib/lsb (symbolic link)

Short Descriptions

ifdown	Stops a network device.
ifup	Initializes a network device.
ipv4-static	Provides the functionality needed to assign a static Internet Protocol (IP) address to a network interface.

12.7. Static Networking Configuration

12.7.1. Creating the Static Network Interface Configuration Files

Which interfaces are brought up and down by the network script depends on the files and directories in the `/etc/sysconfig` hierarchy. This directory should contain a sub-directory for each interface to be configured, such as `ifconfig.xyz`, where “xyz” is a network interface name. Inside this directory would be files defining the attributes to this interface, such as its IP address(es), subnet masks, and so forth.

Note

Udev may assign random Network Card Interface names for some network cards such as `enp2s1`. If you are not sure what your Network Card Interface name is, you can always run **ip l** after you have booted your system. Again, it is important that `ifconfig.xyz` is named after correct Network Card Interface name (e.g. `ifconfig.enp2s1` or `ifconfig.eth0`) or `systemd` will fail to bring up your network interface.

The following command creates a sample `ipv4` file for the `eth0` device:

```
mkdir -pv /etc/sysconfig &&
cd /etc/sysconfig &&
cat > ifconfig.eth0 << "EOF"
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.

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 CLFS-Network-Scripts package has a modular IP assignment format, and creating additional files in the `/lib/services` directory allows other IP assignment methods.

The `GATEWAY` variable should contain the default gateway IP address, if one is present. If not, then comment out the variable entirely.

The `PREFIX` variable needs to contain the number of bits used in the subnet. Each octet in an IP address is 8 bits. If the subnet's netmask is `255.255.255.0`, then it is using the first three octets (24 bits) to specify the network number. If the netmask is `255.255.255.240`, it would be using the first 28 bits. Prefixes longer than 24 bits are commonly used by DSL and cable-based Internet Service Providers (ISPs). In this example (`PREFIX=24`), the netmask is `255.255.255.0`. Adjust the `PREFIX` variable according to your specific subnet.

For more information see the **ifup** man page.

To configure another DHCP Interface, Follow Section 12.8, “DHCPD-6.11.5”.

12.7.2. Configuring the Network Interface at boot

Enabling of the Network Interface configuration is done per interface. To enable Network Interface configuration at boot, run:

```
systemctl enable ifupdown@eth0
```

To disable previously enabled Network Interface configuration at boot, run:

```
systemctl disable ifupdown@eth0
```

To manually start the Network Interface configuration, run:

```
systemctl start ifupdown@eth0
```

Replace eth0 with the correct Network Interface name as described on the beginning of this page.

12.8. DHCPD-6.11.5

The DHCPD package provides a DHCP Client for network configuration.

12.8.1. Installation of DHCPD

If you wish to configure your network to connect to a DHCP server, you will first need to install a DHCP client. CLFS uses the DHCPD package for this.

Prepare DHCPD for compilation:

```
./configure \
  --prefix=/usr \
  --sbindir=/sbin \
  --sysconfdir=/etc \
  --dbdir=/var/lib/dhcpd \
  --libexecdir=/usr/lib/dhcpd
```

Compile the package:

```
make
```

This package does not come with a test suite.

Install the package:

```
make install
```

12.8.2. Creating the DHCP Network Interface Configuration File

The following is an example for the eth0 interface. Refer to the dhcpd.conf man page for more information. This step may be skipped if default behavior of dhcpd is required.

Create the /etc/dhcpd.conf configuration file using the following commands. Adjust appropriately for additional options:

```
cd /etc &&
cat > dhcpd.conf << "EOF"
# dhcpd configuration eth0 interface
# See dhcpd.conf(5) for details.

interface eth0
# dhcpd-run-hooks uses these options.
option subnet_mask, routers, domain_name_servers

# The default timeout for waiting for a DHCP response is 30 seconds
# which may be too long or too short and can be changed here.
timeout 16
EOF
```

To configure another Static Interface, Follow Section 12.7, “Static Networking Configuration”.

12.8.3. Configuring the Network Interface at boot

Enabling of the Network Interface configuration is done per interface. To enable Network Interface configuration at boot, run:

```
systemctl enable dhcpcd@eth0
```

To disable previously enabled Network Interface configuration at boot, run:

```
systemctl disable dhcpcd@eth0
```

To manually start the Network Interface configuration, run:

```
systemctl start dhcpcd@eth0
```

Replace eth0 with the correct Network Interface name as described on the beginning of this page.

12.8.4. Contents of dhcpcd

Installed files: dhcpcd

Short Descriptions

dhcpcd dhcpcd is an implementation of the DHCP client specified in RFC 2131. It gets the host information from a DHCP server and configures the network interface automatically.

Chapter 13. Making the CLFS System Bootable

13.1. Introduction

It is time to make the CLFS system bootable. This chapter discusses building a kernel for the new CLFS system and installing the boot loader so that the CLFS system can be selected for booting at startup.

13.2. Linux-4.9.21

The Linux package contains the Linux kernel.

13.2.1. 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.

Apply the latest Linux sublevel patch:

```
xzcat ../patch-4.9.21.xz | patch -Np1 -i -
```

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.

Note

A good starting place for setting up the kernel configuration is to run **make defconfig**. This will set the base configuration to a good state that takes your current system architecture into account.

Be sure to configure the following options as shown, or the system might not work correctly or boot at all. Refer to `/usr/share/doc/systemd-233/README:`

```
General setup --->
  [*] open by fhandle syscalls (CONFIG_FHANDLE)
  [ ] Auditing support (CONFIG_AUDIT)
  [*] Control Group support (CONFIG_CGROUPS)
Processor type and features --->
  [*] Enable seccomp to safely compute untrusted bytecode (CONFIG_SECCOMP)
Networking support --->
  Networking options --->
    <*> The IPv6 protocol (CONFIG_IPV6)
Device Drivers --->
  Generic Driver Options --->
    ( ) path to uevent helper (CONFIG_UEVENT_HELPER_PATH)
    [*] Maintain a devtmpfs filesystem to mount at /dev (CONFIG_DEVTMPFS)
    [ ] Fallback user-helper invocation for firmware loading (CONFIG_FW_LOADER)
File systems --->
  [*] Inotify support for userspace (CONFIG_FSNOTIFY)
  <*> Kernel automounter version 4 support (also supports v3) (CONFIG_AUTOFS4)
Pseudo filesystems --->
  [*] Tmpfs POSIX Access Control Lists (CONFIG_TMPFS_POSIX_ACL)
  [*] Tmpfs extended attributes (CONFIG_TMPFS_XATTR)
Firmware Drivers --->
  EFI (Extensible Firmware Interface) Support --->
    <*> EFI Variable Support via sysfs (CONFIG_EFI_VARS)
--*-- Enable the block layer ---> (CONFIG_BLOCK)
Partition Types --->
  [*] Advanced partition selection (CONFIG_PARTITION_ADVANCED)
  [*] EFI GUID Partition support (CONFIG_EFI_PARTITION)
Kernel Hacking --->
  [*] Collect scheduler debugging info (CONFIG_SCHED_DEBUG)
  [*] Collect scheduler statistics (CONFIG_SCHEDSTATS)
```

Note

While "The IPv6 Protocol" is not strictly required, it is highly recommended by the Systemd developers. "EFI Variable support" and "EFI GUID Partition support" are for UEFI systems. "Collect scheduler debugging info" and "Collect scheduler statistics" is for systemd-bootchart.

Configure the kernel via a menu-driven interface. CBLFS has some information regarding particular kernel configuration requirements of packages outside of CLFS at <http://cblfs.clfs.org/>:

make menuconfig

If desired, skip kernel configuration by copying the kernel config file, `.config`, from the host system (assuming it is available) to the root directory of the unpacked kernel sources. 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, a configuration file in `/etc/modprobe.d` file may be needed. Information pertaining to modules and kernel configuration is located in the kernel documentation in the `Documentation` directory of the kernel sources tree. Also, `modprobe.d(5)` may be of interest.

Install the modules, if the kernel configuration uses them:

make modules_install

Install the firmware, if the kernel configuration uses them:

make firmware_install

After kernel compilation is complete, additional steps are required to complete the installation. Some files need to be copied to the `/boot` directory.

Issue the following command to install the kernel:

```
cp -v vmlinux /boot/vmlinux-4.9.21
gzip -9 /boot/vmlinux-4.9.21
```

`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. Issue the following command to install the map file:

```
cp -v System.map /boot/System.map-4.9.21
```

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-4.9.21
```

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 do inside the final-system build environment), 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-4.9` 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 a CLFS system as it can cause problems for packages you may wish to build once your base CLFS system is complete.

Also, the headers in the system's `include` directory should *always* be the ones against which Glibc was compiled and should *never* be replaced by headers from a different kernel version.

13.2.2. Contents of Linux

Installed files: `config-[linux-version]`, `clfskernel-[linux-version]`, and `System.map-[linux-version]`
Installed directory: `/lib/modules`

Short Descriptions

<code>config-[linux-version]</code>	Contains all the configuration selections for the kernel
<code>clfskernel-[linux-version]</code>	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.
<code>System.map-[linux-version]</code>	A list of addresses and symbols; it maps the entry points and addresses of all the functions and data structures in the kernel

13.3. Making the CLFS System Bootable via Arcload

Note

This bootloader is for the MIPS based SGI Workstations and Servers.

Earlier, we compiled and installed the Arcload boot loader software in preparation for this step. Now we will configure our system to boot using Arcload. Here is a simple `arc.cf` to use.

```
cat > /boot/arc.cf << "EOF"
append "root=/dev/sda3";
append "console=ttyS0,9600";

CLFS {
  4.9.21 {
    description "4.9.21";
    image system "/4.9.21";
  }

  debug {
    description "Debug Shell";
    append "init=/bin/bash";
  }
}
EOF
```

Now we use **dvhtool** to make the system bootable:

```
dvhtool --unix-to-vh /usr/lib/arcload/sash sash
dvhtool --unix-to-vh /boot/arc.cf arc.cf
dvhtool --unix-to-vh /boot/4.9.21 4.9.21
```

13.4. Making the CLFS System Bootable via Colo

Note

This bootloader is for the MIPS based cobalt servers RaQ, RaQ2, Qube, or the Qube2.

Your shiny new CLFS system is almost complete. One of the last things to do is ensure you can boot it. The instructions below apply only to Cobalt RaQ1/RaQ2/Cube2 servers. Information on “boot loading” for other architectures should be available in the usual resource-specific locations for those architectures.

Boot loading can be a complex area, so a few cautionary words are in order. Be familiar with the current boot loader and any other operating systems present on the hard drive(s) that need to be bootable. Make sure that an emergency boot disk is ready to “rescue” the computer if the computer becomes unusable (un-bootable).

Earlier, we compiled and installed the Cobalt boot loader software in preparation for this step. Now we will configure our system to boot using Colo. Here is a simple `default.colo` to use.

```
cat > /boot/default.colo << "EOF"
#:CoLo:~
#
# load linux
#
lcd 'Booting 4.9.21...'
load vmlinux-4.9.21.gz
execute root=/dev/hda2 console=ttyS0,115200 idel=noprobe
EOF
```

Included in `/usr/lib/colo/examples` are more examples of a `default.colo` file.

The FHS stipulates that the bootloader's configuration file should be symlinked to `/etc/{Bootloader Name}`. To satisfy this requirement for Colo, issue the following command:

```
mkdir -v /etc/colo &&
ln -sv /boot/colo/default.colo /etc/colo
```

Chapter 14. The End

14.1. The End

Well done! The new CLFS 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/clfs-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 CLFS version is installed on the system. Create this file by running:

```
echo GIT-20170803 > /etc/clfs-release
```

14.2. Download Client

The final system build does not install an FTP or HTTP client for downloading files.

Some suggested clients include:

- Curl <http://cblfs.clfs.org/index.php/Curl>
- Inetutils <http://cblfs.clfs.org/index.php/Inetutils>
- LFTP <http://lftp.yar.ru/>
- Links <http://cblfs.clfs.org/index.php/Links>
- Lynx <http://cblfs.clfs.org/index.php/Lynx>
- NcFTP Client <http://cblfs.clfs.org/index.php/Ncftp>
- Wget <http://cblfs.clfs.org/index.php/Wget>
- BASH - A user can use net redirections (if not disabled when building bash in the final system) to download wget or another program.

```
cat > download.sh << "EOF"
#!/bin/bash

WGET_VERSION='1.14'
WGET_HOSTNAME='ftp.gnu.org'
exec {HTTP_FD}<>/dev/tcp/${WGET_HOSTNAME}/80
echo -ne "GET /gnu/wget/wget-${WGET_VERSION}.tar.xz HTTP/1.1\r\nHost: "\
${WGET_HOSTNAME}\r\nUser-Agent: '\
'bash/'${BASH_VERSION}'\r\n\r\n' >&${HTTP_FD}
sed -e '1,/^.$/d' <&${HTTP_FD} >wget-${WGET_VERSION}.tar.xz
EOF
```

- GAWK

```
cat > gawkdl.sh << "EOF"
#!/bin/bash

gawk 'BEGIN {
    NetService = "/inet/tcp/0/mirror.anl.gov/80"
    print "GET /pub/gnu/wget/wget-1.14.tar.xz" |& NetService
    while ((NetService |& getline) > 0)
        print $0
    close(NetService)
}' > binary

gawk '{q=p;p=$0}NR>1{print q}END{ORS = ""; print p}' binary > wget-1.14.tar.xz

rm binary
EOF
```

- PERL with HTTP::Tiny (Included with final system PERL install).

```
cat > download.pl << "EOF"
#!/usr/bin/perl

use HTTP::Tiny;
my $http = HTTP::Tiny->new;
my $response;

$response = $http->mirror('http://ftp.gnu.org/gnu/wget/wget-1.14.tar.xz', 'wget-1.14.tar.xz');
die "Failed!\n" unless $response->{success};
print "Unchanged!\n" if $response->{status} eq '304';
EOF
```

Or use this:

```
perl -MHTTP::Tiny -E 'say HTTP::Tiny->new->get(shift)->{content}' "http://ftp.gnu.org/gnu/wget/wget-1.14.tar.xz"
perl -e 'local $/; $_ = <>; s/\n$//; print' binary > wget-1.14.tar.xz
rm binary
```

- PERL with LWP: Run **cpan** and manually configure the client. Run **install LWP** while in the CPAN shell.

Refer to <http://www.bioinfo-user.org.uk/dokuwiki/doku.php/projects/wgetpl> for wgetpl.

14.3. Rebooting the System

If you built your final system using the boot method, just run **shutdown -r now** to reboot again, using your newly-built kernel instead of the minimal one currently in use. If you chrooted, there are a few more steps.

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 CBLFS 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 CLFS installation. Installing a text mode web browser, such as Lynx, you can easily view the CBLFS website in one virtual

terminal, while building packages in another. The GPM package will also allow you to perform copy/paste actions in your virtual terminals. Lastly, if you are in a situation where static IP configuration does not meet your networking requirements, installing packages such as Dhcpd or PPP at this point might also be useful.

Now that we have said that, let's move on to booting our shiny new CLFS installation for the first time! First exit from the chroot environment:

```
logout
```

Then unmount the virtual file systems:

```
umount ${CLFS}/dev/pts

if [ -h ${CLFS}/dev/shm ]; then
    link=$(readlink ${CLFS}/dev/shm)
    umount -v ${CLFS}/${link}
    unset link
else
    umount -v ${CLFS}/dev/shm
fi

umount ${CLFS}/dev
umount ${CLFS}/proc
umount ${CLFS}/sys
umount ${CLFS}/run
```

Unmount the CLFS file system itself:

```
umount ${CLFS}
```

If multiple partitions were created, unmount the other partitions before unmounting the main one, like this:

```
umount ${CLFS}/usr
umount ${CLFS}/home
umount ${CLFS}
```

Now, reboot the system with:

```
shutdown -r now
```

Assuming the boot loader was set up as outlined earlier, *CLFS GIT-20170803* will boot automatically.

When the reboot is complete, the CLFS system is ready for use and more software may be added to suit your needs.

14.4. What Now?

Thank you for reading this CLFS book. We hope that you have found this book helpful and have learned more about the system creation process.

Now that the CLFS 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 a CLFS system is compiled from source, it is up to you to keep abreast of such reports. There are several online resources that track such reports, some of which are shown below:

- *CERT* (Computer Emergency Response Team)

CERT has a mailing list that publishes security alerts concerning various operating systems and applications. Subscription information is available at <http://www.us-cert.gov/cas/signup.html>.

- Bugtraq

Bugtraq is a full-disclosure computer security mailing list. It publishes newly discovered security issues, and occasionally potential fixes for them. Subscription information is available at <http://www.securityfocus.com/archive>.

- Community Driven Beyond Linux From Scratch

The Community Driven Beyond Linux From Scratch wiki covers installation procedures for a wide range of software beyond the scope of the CLFS Book. CBLFS is designed specifically to work with the CLFS book, and has all the necessary information to continue the builds in the same manner that CLFS uses. This is a community driven project, which means anyone can contribute and provide updates. The CBLFS project is located at <http://cblfs.clfs.org/>.

- CLFS Hints

The CLFS Hints are a collection of educational documents submitted by volunteers in the CLFS community. The hints are available at <http://hints.clfs.org/index.php/>.

- Mailing lists

There are several CLFS 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 (TLPD) is to collaborate on all of the issues of Linux documentation. The TLPD features a large collection of HOWTOs, guides, and man pages. It is located at <http://www.tldp.org/>.

Part VI. Appendices

Appendix A. Acronyms and Terms

ABI	Application Binary Interface
API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
ATA	Advanced Technology Attachment (see IDE)
BIOS	Basic Input/Output System
bles	manipulate a filesystem so that OF will boot from it
BSD	Berkeley Software Distribution
CBLFS	Community Driven Beyond Linux From Scratch
chroot	change root
CLFS	Cross-Compiled Linux From Scratch
CMOS	Complementary Metal Oxide Semiconductor
COS	Class Of Service
CPU	Central Processing Unit
CRC	Cyclic Redundancy Check
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
NPTL	Native POSIX Threading Library
OF	Open Firmware
OSS	Open Sound System
PCH	Pre-Compiled Headers
PID	Process Identifier
PTY	pseudo terminal
QA	Quality Assurance
QOS	Quality Of Service
RAM	Random Access Memory
RPC	Remote Procedure Call
RTC	Real Time Clock
SCO	The Santa Cruz Operation
SATA	Serial ATA
SGR	Select Graphic Rendition
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. Dependencies

Every package built in CLFS 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 CLFS is very important. The purpose of this page is to document the dependencies of each package built in CLFS.

For each package we build, we have listed three 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 last 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.

Acl

Installation depends on: Attr, Bash, Binutils, Coreutils, Glibc, GCC, Gettext, Grep, Libtool, Make, Sed
Test suite depends on: No test suite available
Must be installed before: Coreutils, Gettext, Libcap, Sed, Systemd, Tar, Vim

Attr

Installation depends on: Bash, Binutils, Coreutils, Glibc, GCC, Gettext, Grep, Libtool, Make, Sed
Test suite depends on: No test suite available
Must be installed before: Acl, Coreutils, Gettext, Libcap, Sed, Systemd

Autoconf

Installation depends on: Bash, Coreutils, Gawk, Grep, M4, Make, Perl, Sed, Texinfo
Test suite depends on: Automake, Binutils, Diffutils, Findutils, GCC, Libtool
Must be installed before: Automake

Automake

Installation depends on: Autoconf, Bash, Binutils, Coreutils, Gawk, Grep, M4, Make, Perl, Sed, Texinfo
Test suite depends on: Bison, Bzip2, DejaGNU, Diffutils, Expect, Findutils, Flex, GCC, Gettext, Gzip, Libtool, Tar, XZ Utils. Can also use several other packages that are not installed in CLFS.
Must be installed before: None

Bash

Installation depends on: Bash, Bison, Coreutils, Diffutils, Glibc, Gawk, GCC, Grep, Make, Ncurses, Patch, Readline, Sed, Texinfo
Test suite depends on: None

Must be installed before: None

Bc

Installation depends on: Bash, Binutils, Bison, Coreutils, Glibc, GCC, Grep, Make, Readline

Test suite depends on: Gawk

Must be installed before: None

Binutils

Installation depends on: Bash, Binutils, Coreutils, Diffutils, Glibc, File, Gawk, GCC, Grep, Make, Perl, Sed, Texinfo, Zlib

Test suite depends on: DejaGNU, Expect

Must be installed before: None

Bison

Installation depends on: Bash, Binutils, Coreutils, Glibc, Gawk, GCC, Grep, M4, Make, Sed

Test suite depends on: Diffutils, Findutils, Gawk

Must be installed before: Flex, Kbd, Tar

Bzip2

Installation depends on: Bash, Binutils, Coreutils, Glibc, GCC, Make

Test suite depends on: Diffutils

Must be installed before: None

CLFS-Boot-scripts

Installation depends on: Bash, Coreutils, Make, Sed

Test suite depends on: None

Must be installed before: None

Check

Installation depends on: GCC, Grep, Make, Sed, Texinfo

Test suite depends on: None

Must be installed before: None

Coreutils

Installation depends on: Acl, Attr, Bash, Binutils, Coreutils, Glibc, Gawk, GCC, GMP, Grep, Libcap, Make, Patch, Perl, Sed, Texinfo

Test suite depends on: Diffutils, E2fsprogs, Findutils, Util-linux

Must be installed before: Bash, Diffutils, Findutils, Man

D-Bus

Installation depends on:	Bash, Binutils, Coreutils, Diffutils, Expat, Gawk, GCC, Gettext, Grep, Make, Man, Pkg-config, Sed, Systemd, Texinfo
Test suite depends on:	None
Must be installed before:	None

DejaGNU

Installation depends on:	Bash, Coreutils, Diffutils, GCC, Grep, Make, Sed
Test suite depends on:	None
Must be installed before:	None

DHCPD

Installation depends on:	Bash, Coreutils, GCC, Make, Sed
Test suite depends on:	No test suite available
Must be installed before:	None

Diffutils

Installation depends on:	Bash, Binutils, Coreutils, Glibc, GCC, Grep, Make, Patch, Sed, Texinfo
Test suite depends on:	No test suite available
Must be installed before:	None

Eudev

Installation depends on:	Binutils, Coreutils, Diffutils, Glibc, Gawk, GCC, Grep, Make, Sed
Test suite depends on:	No test suite available
Must be installed before:	Systemd

Expat

Installation depends on:	Bash, Binutils, Coreutils, Diffutils, Glibc, GCC, Grep, Make, Sed
Test suite depends on:	None
Must be installed before:	D-Bus, XML::Parser

Expect

Installation depends on:	Bash, Binutils, Coreutils, Diffutils, Glibc, GCC, Grep, Make, Patch, Sed, Tcl
Test suite depends on:	None
Must be installed before:	None

E2fsprogs

Installation depends on:	Bash, Binutils, Coreutils, Glibc, Gawk, GCC, Gettext, Grep, Gzip, Make, Pkg-config-lite, Sed, Texinfo, Util-linux
---------------------------------	---

Test suite depends on: Bzip2, Diffutils**Must be installed before:** None

File

Installation depends on: Bash, Binutils, Coreutils, Diffutils, Glibc, Gawk, GCC, Grep, Make, Sed, Zlib**Test suite depends on:** No test suite available**Must be installed before:** None

Findutils

Installation depends on: Bash, Binutils, Coreutils, Glibc, GCC, Grep, Make, Sed, Texinfo**Test suite depends on:** DejaGNU, Diffutils, Expect**Must be installed before:** None

Flex

Installation depends on: Bash, Binutils, Coreutils, Glibc, GCC, Grep, M4, Make, Sed, Texinfo**Test suite depends on:** Bison, Diffutils, Gawk**Must be installed before:** IPRoute2, Kbd, Man

Gawk

Installation depends on: Bash, Binutils, Coreutils, Glibc, GCC, GMP, Grep, Make, MPFR, Readline Sed, Texinfo**Test suite depends on:** Diffutils**Must be installed before:** None

Gcc

Installation depends on: Bash, Binutils, Coreutils, Diffutils, Glibc, Findutils, Gawk, GCC, GMP, Grep, ISL, Make, MPFR, Patch, Perl, Sed, Tar, Texinfo**Test suite depends on:** Check, DejaGNU, Expect**Must be installed before:** None

GDBM

Installation depends on: Bash, Binutils, Coreutils, Diffutils, GCC, Grep, Make, Sed**Test suite depends on:** None**Must be installed before:** None

Gettext

Installation depends on: Bash, Binutils, Coreutils, Diffutils, Glibc, Findutils, Gawk, GCC, Grep, Make, Sed, Texinfo**Test suite depends on:** Tar, Tcl**Must be installed before:** Automake

Glibc

Installation depends on:	Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Gettext, Grep, Gzip, Linux Headers, Make, Perl, Sed, Texinfo
Test suite depends on:	None
Must be installed before:	None

GMP

Installation depends on:	Bash, Binutils, Coreutils, Diffutils, Glibc, Gawk, GCC, Grep, M4, Make, Sed, Texinfo
Test suite depends on:	None
Must be installed before:	MPFR, GCC

Gperf

Installation depends on:	Bash, Binutils, Coreutils, Glibc, GCC, Grep, Make, Sed, Texinfo
Test suite depends on:	Intltool
Must be installed before:	Systemd

Grep

Installation depends on:	Bash, Binutils, Coreutils, Glibc, GCC, Grep, Make, Patch, Sed, Texinfo
Test suite depends on:	Diffutils, Gawk
Must be installed before:	Man

Groff

Installation depends on:	Bash, Binutils, Coreutils, Glibc, Gawk, GCC, Grep, Make, Perl Sed, Texinfo
Test suite depends on:	No test suite available
Must be installed before:	Man, Perl

Gzip

Installation depends on:	Bash, Binutils, Coreutils, Glibc, GCC, Grep, Make, Sed, Texinfo
Test suite depends on:	Diffutils
Must be installed before:	Man

lana-Etc

Installation depends on:	Coreutils, Gawk, Make
Test suite depends on:	No test suite available
Must be installed before:	Perl

Intltool

Installation depends on:	Binutils, Coreutils, Glibc, GCC, Make, Perl Sed, XML::Parser
Test suite depends on:	No test suite available

Must be installed before: None

IProute2

Installation depends on: Bash, Binutils, Bison, Coreutils, Glibc, Findutils, Flex, GCC, Make, Linux Headers, Sed

Test suite depends on: No test suite available

Must be installed before: None

IPutils

Installation depends on: Bash, Binutils, Coreutils, Glibc, GCC, Libcap, Make

Test suite depends on: No test suite available

Must be installed before: None

ISL

Installation depends on: Bash, Binutils, Coreutils, Diffutils, Glibc, Gawk, GCC, Grep, GMP, Make, MPC, MPFR, Sed, Texinfo

Test suite depends on: None

Must be installed before: GCC

Kbd

Installation depends on: Bash, Binutils, Check, Coreutils, Glibc, Gawk, GCC, Gzip, Make

Test suite depends on: No test suite available

Must be installed before: None

KMOD

Installation depends on: Bash, Binutils, Bison, Coreutils, Glibc, Flex, Gawk, GCC, Gettext, Gzip, Make, Pkg-config-lite, Sed, XZ Utils, Zlib.

Test suite depends on: No test suite available

Must be installed before: Systemd

Less

Installation depends on: Bash, Binutils, Coreutils, Glibc, GCC, Grep, Make, Ncurses, Sed

Test suite depends on: No test suite available

Must be installed before: None

Libcap

Installation depends on: Attr, Bash, Binutils, Coreutils, Glibc, GCC, Grep, Make

Test suite depends on: No test suite available

Must be installed before: Coreutils, IPutils, Systemd

Libpipeline

Installation depends on:	Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Make, Sed, Texinfo
Test suite depends on:	Check
Must be installed before:	Man-DB

Libtool

Installation depends on:	Bash, Binutils, Coreutils, Diffutils, Glibc, Findutils, Gawk, GCC, Grep, Make, Sed, Texinfo
Test suite depends on:	Autoconf
Must be installed before:	None

Linux Headers

Installation depends on:	Binutils, Coreutils, Findutils, GCC, Grep, Make, Perl, Sed
Test suite depends on:	No test suite available
Must be installed before:	None

Linux Kernel

Installation depends on:	Bash, Binutils, Coreutils, Diffutils, Glibc, Findutils, GCC, Grep, Gzip, Make, KMOD, Ncurses, Perl, Sed
Test suite depends on:	No test suite available
Must be installed before:	None

M4

Installation depends on:	Bash, Binutils, Coreutils, Glibc, Gawk, GCC, Grep, Make, Sed, Texinfo
Test suite depends on:	Diffutils
Must be installed before:	Autoconf, Bison

Make

Installation depends on:	Bash, Binutils, Coreutils, Glibc, GCC, Grep, Make, Sed, Texinfo
Test suite depends on:	Perl, Procps-ng
Must be installed before:	None

Man-DB

Installation depends on:	Bash, Binutils, Bzip2, Coreutils, Glibc, Gawk, GCC, Grep, Groff, Gzip, Less, XZ Utils, Make, Sed
Test suite depends on:	No test suite available
Must be installed before:	D-Bus

Man-Pages

Installation depends on:	Bash, Coreutils, Make
---------------------------------	-----------------------

Test suite depends on: No test suite available
Must be installed before: None

MPC

Installation depends on: Bash, Binutils, Coreutils, Diffutils, Glibc, Gawk, GCC, Grep, GMP, Make, MPFR, Sed, Texinfo
Test suite depends on: None
Must be installed before: GCC

MPFR

Installation depends on: Bash, Binutils, Coreutils, Diffutils, Glibc, Gawk, GCC, Grep, GMP, Make, Sed, Texinfo
Test suite depends on: None
Must be installed before: Gawk, GCC

Ncurses

Installation depends on: Bash, Binutils, Coreutils, Diffutils, Glibc, Gawk, GCC, Grep, Make, Pkg-config-lite, Sed
Test suite depends on: No test suite available
Must be installed before: Bash, GRUB, Inetutils, Less, Procps-ng, Psmisc, Readline, Texinfo, Util-linux, Vim

Patch

Installation depends on: Bash, Binutils, Coreutils, Glibc, GCC, Grep, Make, Sed
Test suite depends on: No test suite available
Must be installed before: None

Perl

Installation depends on: Bash, Binutils, Bzip2, Coreutils, Glibc, Gawk, GCC, Grep, Make, Sed
Test suite depends on: Gzip, Iana-Etc, Procps-ng, Tar
Must be installed before: Autoconf

Pkg-config-lite

Installation depends on: Bash, Binutils, Coreutils, Diffutils, Glibc, Gawk, GCC, Grep, Make, Sed
Test suite depends on: None
Must be installed before: E2fsprogs, Systemd, Util-linux

Procps-ng

Installation depends on: Bash, Binutils, Coreutils, Glibc, GCC, Make, Ncurses
Test suite depends on: No test suite available
Must be installed before: None

Psmisc

Installation depends on: Bash, Binutils, Coreutils, Glibc, GCC, Grep, Make, Ncurses , Sed
Test suite depends on: No test suite available
Must be installed before: None

Readline

Installation depends on: Bash, Binutils, Coreutils, Glibc, GCC, Grep, Make, Ncurses, Patch, Sed, Texinfo
Test suite depends on: No test suite available
Must be installed before: Bash, Gawk

Sed

Installation depends on: Acl, Attr, Bash, Binutils, Coreutils, Glibc, GCC, Grep, Make, Sed, Texinfo
Test suite depends on: Diffutils, Gawk
Must be installed before: E2fsprogs, File, Libtool, Shadow

Shadow

Installation depends on: Acl, Attr, Bash, Binutils, Coreutils, Diffutils, Glibc, Findutils, Gawk, GCC, Gettext, Grep, Make, Sed
Test suite depends on: No test suite available
Must be installed before: None

Systemd

Installation depends on: Acl, Attr, Bash, Binutils, Coreutils, E2fsprogs, Glibc, Findutils, Gawk, GCC, GPerf, Grep, Intltool, Libcap, Make, Perl, Pkg-config, Sed, Util-linux, XML::Parser
Test suite depends on: No test suite available
Must be installed before: D-Bus

Tar

Installation depends on: Acl, Attr, Bash, Binutils, Bison, Coreutils, Glibc, GCC, Grep, Make, Sed, Texinfo
Test suite depends on: Diffutils, Findutils, Gawk, Gzip
Must be installed before: None

Tcl

Installation depends on: Bash, Binutils, Coreutils, Diffutils, Glibc, GCC, Grep, Make, Sed
Test suite depends on: None
Must be installed before: None

Texinfo

Installation depends on: Bash, Binutils, Coreutils, Glibc, Gawk, GCC, Grep, Make, Ncurses, Sed

Test suite depends on: Diffutils, Gzip
Must be installed before: None

Util-linux

Installation depends on: Bash, Binutils, Coreutils, Glibc, GCC, Grep, Make, Ncurses, Pkg-config-lite, Sed, Texinfo, Zlib
Test suite depends on: No test suite available
Must be installed before: E2fsprogs, Systemd

Vim

Installation depends on: Acl, Attr, Bash, Binutils, Coreutils, Diffutils, Glibc, Findutils, Gawk, GCC, Gettext, Grep, Make, Ncurses, Perl, Sed
Test suite depends on: Gzip
Must be installed before: None

XML::Parser

Installation depends on: Coreutils, Expat, Make, Perl
Test suite depends on: None
Must be installed before: Intltool

XZ Utils

Installation depends on: Bash, Binutils, Coreutils, Diffutils, Glibc, Findutils, Gawk, GCC, Grep, Make, Sed
Test suite depends on: None
Must be installed before: None

Zlib

Installation depends on: Bash, Binutils, Coreutils, Glibc, GCC, Make, Sed
Test suite depends on: None
Must be installed before: File, KMOD, Util-linux

Appendix C. Mips Dependencies

This page contains dependency information for packages specific to Mips.

Arcload

Installation depends on: Binutils, Coreutils, Dvhtool, GCC, Make, Sed
Test suite depends on: None
Must be installed before: None

Colo

Installation depends on: Binutils, Coreutils, GCC, Gzip, Make
Test suite depends on: None
Must be installed before: None

Dvhtool

Installation depends on: Binutils, Coreutils, GCC, Make
Test suite depends on: None
Must be installed before: None

Appendix D. Package Rationale

CLFS includes many packages, a number of which might not necessarily be required for a "minimal" system, but still considered very useful. The purpose of this page is to list the reasoning for each package's inclusion in the book.

- Acl

The Acl package allows usage and setting of POSIX Access Control Lists. It can be used by several other packages in CLFS, such as Coreutils and Systemd.

- Attr

Attr allows setting and viewing extended attributes of filesystem objects. It is required by Systemd.

- Autoconf

The Autoconf package contains programs for producing shell scripts that can automatically configure source code. This is useful for software developers, as well as anyone who wants to install packages that don't come with a configure script, such as some of the packages in CBLFS.

- Automake

The Automake package contains programs for generating Makefiles for use with Autoconf. This can be useful to software developers.

- Bash

This package contains the Bourne-Again SHell. A shell is an important component of a Linux system, as there must be some way of allowing the users to enter commands.

- Bc

This package contains a precision calculator. The Linux kernel uses Bc to render the timeconst header.

- Binutils

This package contains programs for handling object files. The programs in this package are needed for compiling most of the packages in CLFS.

- Bison

This package contains programs that are required by several packages in CLFS.

- Bzip2

The programs in this package are useful for compressing files to reduce size. They are also needed to uncompress tarballs for many CLFS packages.

- CLFS-Boot-scripts

This package contains a number of scripts that run at boottime, performing essential tasks such as mounting/checking filesystems and starting the network interface.

- Check

This package contains a test harness for other programs. It is used for some packages' test suites.

- Coreutils

This package contains many basic command-line file-management tools, required for installation of every package in CLFS.

- D-Bus

D-Bus is a message bus system, which allows applications to communicate to each other. It is used by Systemd.

- DejaGNU

This package is needed for the test suites of several packages, especially GCC and Binutils.

- DHCPD

This package allows for automatic configuration of network interfaces from a DHCP server. It (or some other package providing a DHCP client is needed to connect to a DHCP server.

- Diffutils

This package contains programs to compare files, and can also be used to create patches. It is required by the installation procedures of many CLFS packages, and used by many packages' test suites.

- Eudev

This is a package that allows for dynamic creation of device nodes. It is a fork of Udev, which is now part of Systemd. It is still used for the "Boot" method in the temp-system, as Systemd is not needed there.

- Expect

This package is needed for the test suites for several packages.

- E2fsprogs

The programs in this package are used for the creation and maintenance of ext2/3/4 filesystems.

- File

This package contains a program that determines the type of a given file. It is needed by some CLFS packages.

- Findutils

This package contains programs for finding files based on certain criteria, and optionally performing commands on them. These programs are used by the installation procedures of many CLFS packages.

- Flex

This package contains a tool for generating text scanners. It is used by multiple packages in CLFS

- Gawk

This package contains programs for manipulating text files, using the AWK language. It is used by the installation procedures of many packages in CLFS.

- Gcc

This package contains a C compiler, which is required to compile most of the packages in CLFS.

- GDBM

This package contains the GNU Database Manager library. Man-DB requires either GDBM or Berkeley DB, though it prefers GDBM.

- Gettext

A tool that allows programmers to easily implement i18n (internationalization) in their programs. It is a required dependency for a number of packages

- Glibc

Any dynamically-linked C program (nearly every package in CLFS has these) needs a C library to compile and run.

- GMP

This package is required by GCC.

- Gperf

This package is required by Systemd.

- Grep

This package contains programs for searching for text in files. These programs are required by many packages in CLFS.

- Groff

This package is required by Man-DB.

- Gzip

Useful for compressing files to reduce size. It is also needed to uncompress tarballs for many CLFS packages

- Iana-Etc

This package provides the `/etc/services` and `/etc/protocols` files. These files map port names to port numbers as well as protocol names to their corresponding numbers. These files are essential for many network based programs to work properly.

- Intltool

This package is required by Systemd.

- IProute2

This package contains programs for administering network interfaces.

- IPutils

This package contains several basic network-management tools.

- ISL

This package is required by GCC for GRAPHITE optimizations.

- Kbd

Contains keytable files and keyboard utilities compatible with the Linux kernel. These can be used to change the display font and keyboard layout.

- Kmod

This package contains programs that assist in loading and unloading kernel modules.

- Less

A program that lets you view text files one page at a time. It is also used by Man-DB for displaying manpages.

- Libcap

This package is required by Systemd.

- 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

The Libtool package contains the GNU generic library support script. It is used by some CLFS packages.

- Linux Headers

This package consists of sanitized headers from the Linux Kernel. These headers are required for Glibc to compile.

- Linux Kernel

The Linux operating system.

- M4

This package contains a macro processor. It is required by several CLFS packages, including Bison.

- Make

This is required for installation of most CLFS packages

- Man-DB

This package contains programs for finding and viewing man pages, and has superior internationalization capabilities compared to the Man package.

- Man-Pages

A number of useful manpages, not supplied by other packages

- MPC

This package is required by GCC.

- MPFR

This package is required by GCC.

- Ncurses

Needed by several packages in CLFS, such as Vim, Bash, and Less

- Patch

Used for applying patches in several CLFS packages

- Perl

The Perl package contains the Practical Extraction and Report Language. It is required by several CLFS packages.

- Pkg-config-lite

Several packages in CLFS, and many others outside of CLFS, use **pkg-config** to locate dependencies.

- Procps-ng

Provides a number of small, useful utilities that give information about the `/proc` filesystem.

- Psmisc

Provides more utilities that give information about the `/proc` filesystem.

- Readline

The Readline library provides a set of functions for use by applications that allow users to edit command lines as they are typed in. This is essential for input in programs like **bash** to work properly.

- Sed

This package contains a stream editor. It is used in the installation procedures of most CLFS packages.

- Shadow

This package contains programs that assist in the administration of users and groups, and passwords.

- Systemd

Systemd provides the init daemon for the system, as well as Udev, which dynamically creates device nodes.

- Tar

Required to unpack the tar archives in which all CLFS packages are distributed

- Tcl

Needed for the test suites of several packages

- Texinfo

This package contains programs for viewing, installing and converting info pages. It is used in the installation procedures of many CLFS packages.

- Util-linux

The Util-linux package contains miscellaneous utility programs. Among them are utilities for handling file systems, consoles, partitions, and messages. It also includes libraries that are required by E2fsprogs.

- Vim

The Vim package contains a text editor. Users may substitute Nano, Joe, Emacs, or whatever other editor they prefer.

- XML::Parser

This Perl module is required by Intltool.

- XZ Utils

Useful for compressing files to reduce size. Also needed to uncompress tarballs for many CLFS packages

- Zlib

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

Appendix E. Package Rationale - MIPS

This is the explanation for the inclusion of MIPS-specific packages.

- ARCLoad

An SGI Multi-bootloader. Able to bootload many different SGI Systems.

- Colo

A replacement bootloader for the Cobalt MIPS based Raq/Qube? servers.

- DVHTool

Dvhtool is the tool responsible for writing MIPS kernel(s) into the SGI volume header.

Appendix F. Open Publication License

v1.0, 8 June 1999

I. REQUIREMENTS ON BOTH UNMODIFIED AND MODIFIED VERSIONS

The Open Publication works may be reproduced and distributed in whole or in part, in any medium physical or electronic, provided that the terms of this license are adhered to, and that this license or an incorporation of it by reference (with any options elected by the author(s) and/or publisher) is displayed in the reproduction.

Proper form for an incorporation by reference is as follows:

Copyright © <year> by <author's name or designee>. This material may be distributed only subject to the terms and conditions set forth in the Open Publication License, vX.Y or later (the latest version is presently available at <http://www.opencontent.org/openpub/>).

The reference must be immediately followed with any options elected by the author(s) and/or publisher of the document (see section VI).

Commercial redistribution of Open Publication-licensed material is permitted.

Any publication in standard (paper) book form shall require the citation of the original publisher and author. The publisher and author's names shall appear on all outer surfaces of the book. On all outer surfaces of the book the original publisher's name shall be as large as the bridgehead of the work and cited as possessive with respect to the bridgehead.

II. COPYRIGHT

The copyright to each Open Publication is owned by its author(s) or designee.

III. SCOPE OF LICENSE

The following license terms apply to all Open Publication works, unless otherwise explicitly stated in the document.

Mere aggregation of Open Publication works or a portion of an Open Publication work with other works or programs on the same media shall not cause this license to apply to those other works. The aggregate work shall contain a notice specifying the inclusion of the Open Publication material and appropriate copyright notice.

SEVERABILITY. If any part of this license is found to be unenforceable in any jurisdiction, the remaining portions of the license remain in force.

NO WARRANTY. Open Publication works are licensed and provided "as is" without warranty of any kind, express or implied, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose or a warranty of non-infringement.

IV. REQUIREMENTS ON MODIFIED WORKS

All modified versions of documents covered by this license, including translations, anthologies, compilations and partial documents, must meet the following requirements:

1. The modified version must be labeled as such.
2. The person making the modifications must be identified and the modifications dated.

3. Acknowledgement of the original author and publisher if applicable must be retained according to normal academic citation practices.
4. The location of the original unmodified document must be identified.
5. The original author's (or authors') name(s) may not be used to assert or imply endorsement of the resulting document without the original author's (or authors') permission.

V. GOOD-PRACTICE RECOMMENDATIONS

In addition to the requirements of this license, it is requested from and strongly recommended of redistributors that:

1. If you are distributing Open Publication works on hardcopy or CD-ROM, you provide email notification to the authors of your intent to redistribute at least thirty days before your manuscript or media freeze, to give the authors time to provide updated documents. This notification should describe modifications, if any, made to the document.
2. All substantive modifications (including deletions) be either clearly marked up in the document or else described in an attachment to the document.
3. Finally, while it is not mandatory under this license, it is considered good form to offer a free copy of any hardcopy and CD-ROM expression of an Open Publication-licensed work to its author(s).

VI. LICENSE OPTIONS

The author(s) and/or publisher of an Open Publication-licensed document may elect certain options by appending language to the reference to or copy of the license. These options are considered part of the license instance and must be included with the license (or its incorporation by reference) in derived works.

A. To prohibit distribution of substantively modified versions without the explicit permission of the author(s). "Substantive modification" is defined as a change to the semantic content of the document, and excludes mere changes in format or typographical corrections.

To accomplish this, add the phrase 'Distribution of substantively modified versions of this document is prohibited without the explicit permission of the copyright holder.' to the license reference or copy.

B. To prohibit any publication of this work or derivative works in whole or in part in standard (paper) book form for commercial purposes is prohibited unless prior permission is obtained from the copyright holder.

To accomplish this, add the phrase 'Distribution of the work or derivative of the work in any standard (paper) book form is prohibited unless prior permission is obtained from the copyright holder.' to the license reference or copy.

OPEN PUBLICATION POLICY APPENDIX

(This is not considered part of the license.)

Open Publication works are available in source format via the Open Publication home page at <http://works.opencontent.org/>.

Open Publication authors who want to include their own license on Open Publication works may do so, as long as their terms are not more restrictive than the Open Publication license.

If you have questions about the Open Publication License, please contact David Wiley at dw@opencontent.org, and/or the Open Publication Authors' List at opal@opencontent.org, via email.

To **subscribe** to the Open Publication Authors' List: Send E-mail to opal-request@opencontent.org with the word "subscribe" in the body.

To **post** to the Open Publication Authors' List: Send E-mail to opal@opencontent.org or simply reply to a previous post.

To **unsubscribe** from the Open Publication Authors' List: Send E-mail to opal-request@opencontent.org with the word "unsubscribe" in the body.

Index

Packages

Acl: 152
 Arcload: 241
 configuring: 270
 Attr: 151
 Autoconf: 184
 Automake: 185
 Bash: 187
 temporary system: 63
 Bc: 189
 boot: 86
 Binutils: 145
 cross tools: 43
 temporary system: 59
 Bison: 144
 Boot-scripts: 87
 Bzip2: 177
 temporary system: 65
 Check: 66
 Colo: 238
 boot: 100
 boot, configuring: 106
 configuring: 270
 Coreutils: 168
 temporary system: 67
 D-Bus: 226
 DejaGNU: 120
 DHCPCD: 263
 Diffutils: 190
 temporary system: 68
 Dvhtool: 240
 E2fsprogs: 165
 boot: 89
 Eudev: 95
 Expat: 214
 Expect: 119
 File: 191
 cross tools: 34
 temporary system: 69
 Findutils: 193
 temporary system: 70
 Flex: 143
 Gawk: 192
 temporary system: 71
 GCC: 148
 cross tools, final: 50
 cross tools, static: 45
 temporary system: 60
 GDBM: 179
 Gettext: 194
 temporary system: 72
 Glibc: 128
 cross tools: 48
 GMP: 137
 cross tools: 39
 temporary system: 54
 Gperf: 196
 Grep: 197
 temporary system: 73
 Groff: 198
 Gzip: 202
 temporary system: 74
 Iana-Etc: 173
 Intltool: 216
 IPRoute2: 175
 IPutils: 204
 ISL: 141
 cross tools: 42
 temporary system: 57
 Kbd: 205
 Kmod: 217
 boot: 90
 Less: 201
 Libcap: 154
 Libpipeline: 207
 Libtool: 174
 Linux: 266
 boot: 97
 Linux Headers: 126
 cross tools: 35
 M4: 136
 cross tools: 36
 Make: 211
 temporary system: 75
 Man-DB: 208
 Man-pages: 127
 MPC: 140
 cross tools: 41
 temporary system: 56
 MPFR: 139
 cross tools: 40

temporary system: 55
 Ncurses: 157
 cross tools: 37
 temporary system: 62
 Network-Scripts: 260
 Patch: 219
 temporary system: 76
 Perl: 180
 temporary tools: 125
 Pkg-config-lite: 156
 cross tools: 38
 Procps-ng: 163
 Psmisc: 220
 Readline: 183
 Sed: 155
 temporary system: 77
 Shadow: 159
 boot: 91
 configuring: 160
 Systemd: 221
 configuring: 223
 usage: 243
 Sysvinit: 92
 configuring: 92
 Tar: 228
 temporary system: 78
 Tcl: 118
 Texinfo: 229
 temporary system: 79
 Udev
 usage: 246
 Util-linux: 230
 temporary system: 80
 Util-linux: 162
 Vim: 235
 temporary system: 81
 XML::Parser: 215
 XZ Utils: 212
 temporary system: 83
 Zlib: 142
 temporary system: 58

Programs
 a2p: 180, 181
 accessdb: 208, 210
 aclocal: 185, 185
 aclocal-1.15: 185, 185
 addftinfo: 198, 198
 addpart: 230, 231
 addr2line: 145, 146
 afmtodit: 198, 198
 agetty: 230, 231
 apropos: 208, 210
 ar: 145, 146
 as: 145, 146
 ata_id: 95, 96
 attr: 151, 151
 autoconf: 184, 184
 autoheader: 184, 184
 autom4te: 184, 184
 automake: 185, 185
 automake-1.15: 185, 185
 autopoint: 194, 194
 autoreconf: 184, 184
 autoscan: 184, 184
 autoupdate: 184, 184
 awk: 192, 192
 badblocks: 165, 166
 base64: 168, 169
 basename: 168, 169
 bash: 187, 188
 bashbug: 187, 188
 bc: 189, 189
 bigram: 193, 193
 bison: 144, 144
 blkdiscard: 230, 231
 blkid: 230, 231
 blockdev: 230, 231
 bootctl: 221, 224
 bootlogd: 92, 93
 bridge: 175, 175
 bunzip2: 177, 177
 busctl: 221, 224
 bzip2: 177, 177
 bzip2recover: 177, 178
 bzcat: 177, 177
 bzcmp: 177, 178
 bzdiff: 177, 178
 bzegrep: 177, 178
 bzfgrep: 177, 178
 bzgrep: 177, 178
 bzip2: 177, 178
 bzip2recover: 177, 178
 bzless: 177, 178
 bzip2more: 177, 178
 c++: 148, 149

c++filt: 145, 146
 c2ph: 180, 181
 cal: 230, 231
 capsh: 154, 154
 captinfo: 157, 158
 cat: 168, 169
 catchsegv: 128, 133
 catman: 208, 210
 cc: 148, 149
 cdrom_id: 95, 96
 cfdisk: 230, 231
 chacl: 152, 152
 chage: 159, 161
 chattr: 165, 166
 chcon: 168, 169
 chcpu: 230, 231
 checkmk: 66, 66
 chem: 198, 198
 chfn: 159, 161
 chpasswd: 159, 161
 chgrp: 168, 169
 chmod: 168, 169
 chown: 168, 169
 chpasswd: 159, 161
 chroot: 168, 169
 chrt: 230, 231
 chsh: 159, 161
 chvt: 205, 206
 cksum: 168, 169
 clear: 157, 158
 clfskernel-[linux-version]: 266, 269
 clockdiff: 204, 204
 cmp: 190, 190
 code: 193, 193
 col: 230, 231
 colert: 230, 231
 collect: 95, 96
 colo-chain.elf: 238, 238
 colrm: 230, 231
 column: 230, 231
 comm: 168, 169
 compile: 185, 185
 compile_et: 165, 166
 config.charset: 194, 194
 config.guess: 185, 185
 config.rpath: 194, 194
 config.sub: 185, 185
 config_data: 180, 181
 corelist: 180, 181
 cp: 168, 169
 cpan: 180, 181
 cpan2dist: 180, 181
 cpanp: 180, 181
 cpanp-run-perl: 180, 181
 cpp: 148, 149
 create_floppy_devices: 95, 96
 csplit: 168, 169
 ctrlaltdel: 230, 231
 ctstat: 175, 175
 cut: 168, 169
 date: 168, 169
 dbus-cleanup-sockets: 226, 226
 dbus-daemon: 226, 226
 dbus-launch: 226, 226
 dbus-monitor: 226, 226
 dbus-send: 226, 227
 dbus-uuidgen: 226, 226
 dc: 189, 189
 dd: 168, 169
 deallocvt: 205, 206
 debugfs: 165, 166
 delpart: 230, 231
 depcomp: 185, 185
 depmod: 217, 217
 df: 168, 169
 diff: 190, 190
 diff3: 190, 190
 dir: 168, 169
 dircolors: 168, 170
 dirname: 168, 170
 dmesg: 230, 231, 230, 231
 du: 168, 170
 dumpe2fs: 165, 166
 dumpkeys: 205, 206
 dvhtool: 240, 240
 e2freefrag: 165, 166
 e2fsck: 165, 166
 e2fsck-lcd: 238, 238
 e2image: 165, 166
 e2initrd_helper: 165, 166
 e2label: 165, 166
 e2undo: 165, 166
 e4defrag: 165, 166
 echo: 168, 170

edd_id: 95, 96
 efm_filter.pl: 235, 236
 efm_perl.pl: 235, 236
 egrep: 197, 197
 elfedit: 145, 146
 enc2xs: 180, 181
 env: 168, 170
 envsubst: 194, 194
 eqn: 198, 198
 eqn2graph: 198, 198
 ex: 235, 236
 expand: 168, 170
 expect: 119, 119
 expiry: 159, 161
 expr: 168, 170
 factor: 168, 170
 faillog: 159, 161
 fallocate: 230, 231
 false: 168, 170
 fdformat: 230, 231
 fdisk: 230, 231
 fgconsole: 205, 206
 fgrep: 197, 197
 file: 191, 191
 filefrag: 165, 166
 find: 193, 193
 find2perl: 180, 181
 findfs: 230, 231
 findmnt: 230, 231
 firmware.sh: 95, 96
 flex: 143, 143
 flex++: 143, 143
 flock: 230, 231
 fmt: 168, 170
 fold: 168, 170
 frcode: 193, 193
 free: 163, 164
 fsck: 230, 231
 fsck.cramfs: 230, 232
 fsck.ext2: 165, 166
 fsck.ext3: 165, 166
 fsck.ext4: 165, 166
 fsck.ext4dev: 165, 166
 fsck.minix: 230, 232
 fsfreeze: 230, 232
 fstab-decode: 92, 93
 fstab_import: 95, 96

fstrim: 230, 232
 fuser: 220, 220
 g++: 148, 149
 gawk: 192, 192
 gawk-4.1.4: 192, 192
 gcc: 148, 149
 gcov: 148, 149
 gcov-tool: 148, 149
 gdbmtool: 179, 179
 gdbm_dump: 179, 179
 gdbm_load: 179, 179
 gdiffmk: 198, 198
 gencat: 128, 133
 genl: 175, 175
 getcap: 154, 154
 getconf: 128, 133
 getent: 128, 133
 getfacl: 152, 153
 getfattr: 151, 151
 getkeycodes: 205, 206
 getopt: 230, 232
 getpcaps: 154, 154
 gettext: 194, 194
 gettext.sh: 194, 194
 gettextize: 194, 194
 gpasswd: 159, 161
 gperf: 196, 196
 gprof: 145, 146
 grap2graph: 198, 198
 grcat: 192, 192
 grep: 197, 197
 gm: 198, 198
 grodvi: 198, 198
 groff: 198, 198
 groffer: 198, 198
 grog: 198, 199
 grolbp: 198, 199
 grolj4: 198, 199
 grops: 198, 199
 grotty: 198, 199
 groupadd: 159, 161
 groupdel: 159, 161
 groupmems: 159, 161
 groupmod: 159, 161
 groups: 168, 170
 grpck: 159, 161
 grpconv: 159, 161

grpunconv: 159, 161
 gunzip: 202, 202
 gzexe: 202, 202
 gzip: 202, 202
 h2ph: 180, 181
 h2xs: 180, 181
 halt: 221, 224
 halt: 92, 93
 head: 168, 170
 hexdump: 230, 232
 hostid: 168, 170
 hostname: 168, 170
 hostname: 194, 194
 hostnamectl: 221, 224
 hpftodit: 198, 199
 hwclock: 230, 232
 iconv: 128, 133
 iconvconfig: 128, 133
 id: 168, 170
 ifcfg: 175, 175
 ifnames: 184, 184
 ifstat: 175, 175
 igawk: 192, 192
 indxbib: 198, 199
 info: 229, 229
 infocmp: 157, 158
 infokey: 229, 229
 infotocap: 157, 158
 init: 221, 224
 init: 92, 93
 insmod: 217, 218
 install: 168, 170
 install-info: 229, 229
 install-sh: 185, 186
 instmodsh: 180, 181
 intltool-extract: 216, 216
 intltool-merge: 216, 216
 intltool-prepare: 216, 216
 intltool-update: 216, 216
 intltoolize: 216, 216
 ionice: 230, 232
 ip: 175, 175
 ipcmk: 230, 232
 ipcrm: 230, 232
 ipcs: 230, 232
 isosize: 230, 232
 join: 168, 170
 journalctl: 221, 224
 json_pp: 180, 181
 kbdinfo: 205, 206
 kbdrate: 205, 206
 kbd_mode: 205, 206
 kernel-install: 221, 224
 kill: 230, 232
 killall: 220, 220
 killall5: 92, 93
 kmod: 217, 218
 last: 230, 232
 lastb: 230, 232
 lastlog: 159, 161
 ld: 145, 146
 ld.bfd: 145, 146
 ld.gold: 145, 146
 ldattach: 230, 232
 ldconfig: 128, 133
 ldd: 128, 133
 lddlibc4: 128, 133
 less: 201, 201
 less.sh: 235, 236
 lessecho: 201, 201
 lesskey: 201, 201
 lex: 143, 143
 lexgrog: 208, 210
 libnetcfg: 180, 181
 libtool: 174, 174
 libtoolize: 174, 174
 link: 168, 170
 lkbib: 198, 199
 ln: 168, 170
 lnstat: 175, 176
 loadkeys: 205, 206
 loadunimap: 205, 206
 locale: 128, 133
 localectl: 221, 224
 localedef: 128, 133
 locate: 193, 193
 logger: 230, 232
 login: 159, 161
 loginctl: 221, 224
 logname: 168, 170
 logoutd: 159, 161
 logsave: 165, 166
 look: 230, 232
 lookbib: 198, 199

losetup: 230, 232	mkswap: 230, 232
ls: 168, 170	mktemp: 168, 170
lsattr: 165, 166	mk_cmds: 165, 166
lsblk: 230, 232	mmroff: 198, 199
lscpu: 230, 232	modinfo: 217, 218
lslocks: 230, 232	modprobe: 217, 218
lsmod: 217, 218	more: 230, 232
lzcat: 212, 212	mount: 230, 232
lzcmp: 212, 212	mountpoint: 230, 232
lzdiff: 212, 212	msgattrib: 194, 195
lzegrep: 212, 212	msgcat: 194, 195
lzfgrep: 212, 212	msgcmp: 194, 195
lzgrep: 212, 213	msgcomm: 194, 195
lzless: 212, 213	msgconv: 194, 195
lzma: 212, 213	msgen: 194, 195
lzmadec: 212, 213	msgexec: 194, 195
lzmainfo: 212, 213	msgfilter: 194, 195
lzmore: 212, 213	msgfmt: 194, 195
m4: 136, 136	msggrep: 194, 195
machinectl: 221, 224	msginit: 194, 195
make: 211, 211	msgmerge: 194, 195
makedb: 128, 133	msgunfmt: 194, 195
makeinfo: 229, 229	msguniq: 194, 195
man: 208, 210	mtrace: 128, 133
mandb: 208, 210	mv: 168, 170
manpath: 208, 210	mve.awk: 235, 236
mapscrn: 205, 206	namei: 230, 232
mcookie: 230, 232	ncursesw6-config: 157, 158
md5sum: 238, 239	neqn: 198, 199
md5sum: 168, 170	newgrp: 159, 161
mdate-sh: 185, 186	newusers: 159, 161
mesg: 230, 232	ngettext: 194, 195
missing: 185, 186	nice: 168, 170
mkdir: 168, 170	nl: 168, 170
mke2fs: 165, 166	nm: 145, 146
mkfifo: 168, 170	nohup: 168, 170
mkfs: 230, 232	nologin: 159, 161
mkfs.bfs: 230, 232	nproc: 168, 170
mkfs.cramfs: 230, 232	nroff: 198, 199
mkfs.ext2: 165, 166	nscd: 128, 133
mkfs.ext3: 165, 166	nsenter: 230, 232
mkfs.ext4: 165, 166	nstat: 175, 176
mkfs.ext4dev: 165, 166	numfmt: 168, 170
mkfs.minix: 230, 232	objcopy: 145, 146
mkinstalldirs: 185, 186	objdump: 145, 146
mklost+found: 165, 167	od: 168, 170
mknod: 168, 170	oldfind: 193, 193

openvt: 205, 206
 panel: 238, 238
 partx: 230, 233
 passwd: 159, 161
 paste: 168, 171
 patch: 219, 219
 pathchk: 168, 171
 path_id: 95, 96
 pcprofiledump: 128, 133
 pdfroff: 198, 199
 pdftexi2dvi: 229, 229
 peekfd: 220, 220
 perl: 180, 181
 perl5.26.0: 180, 181
 perlbug: 180, 181
 perldoc: 180, 182
 perlvp: 180, 182
 perlthanks: 180, 182
 pfbtops: 198, 199
 pg: 230, 233
 pgawk: 192, 192
 pgawk-4.1.4: 192, 192
 pgrep: 163, 164
 pic: 198, 199
 pic2graph: 198, 199
 piconv: 180, 182
 pidof: 163, 164
 ping: 204, 204
 ping: 204, 204
 ping6: 204, 204
 pinky: 168, 171
 pivot_root: 230, 233
 pkg-config: 156, 156
 pkill: 163, 164
 pl2pm: 180, 182
 pldd: 128, 133
 pltags.pl: 235, 236
 pmap: 163, 164
 pod2html: 180, 182
 pod2latex: 180, 182
 pod2man: 180, 182
 pod2text: 180, 182
 pod2usage: 180, 182
 podchecker: 180, 182
 podselect: 180, 182
 post-grohtml: 198, 199
 poweroff: 221, 224
 poweroff: 92, 93
 pr: 168, 171
 pre-grohtml: 198, 199
 preconv: 198, 199
 printenv: 168, 171
 printf: 168, 171
 prlimit: 230, 233
 prove: 180, 182
 prtstat: 220, 220
 ps: 163, 164
 psed: 180, 182
 psfaddtable: 205, 206
 psfgettable: 205, 206
 psfstriptime: 205, 206
 psfxtable: 205, 206
 pstree: 220, 220
 pstree.x11: 220, 220
 pstruct: 180, 182
 ptar: 180, 182
 ptardiff: 180, 182
 ptargrep: 180, 182
 ptx: 168, 171
 putlcd: 238, 238
 pwcat: 192, 192
 pwck: 159, 161
 pwconv: 159, 161
 pwd: 168, 171
 pwdx: 163, 164
 pwunconv: 159, 161
 py-compile: 185, 186
 ranlib: 145, 146
 raw: 230, 233
 rdisc: 204, 204
 readelf: 145, 146
 readlink: 168, 171
 readprofile: 230, 233
 realpath: 168, 171
 reboot: 221, 224
 reboot: 92, 93
 recode-sr-latin: 194, 195
 ref: 235, 236
 refer: 198, 199
 rename: 230, 233
 renice: 230, 233
 reset: 157, 158
 resize2fs: 165, 167
 resizecons: 205, 206

resizepart: 230, 233	sg: 159, 161
rev: 230, 233	sh: 187, 188
rm: 168, 171	sha1sum: 168, 171
rmdir: 168, 171	sha224sum: 168, 171
rmmod: 217, 218	sha256sum: 168, 171
rmt: 228, 228	sha384sum: 168, 171
roff2dvi: 198, 199	sha512sum: 168, 171
roff2html: 198, 199	shasum: 180, 182
roff2pdf: 198, 199	showconsolefont: 205, 206
roff2ps: 198, 199	showkey: 205, 206
roff2text: 198, 199	shred: 168, 171
roff2x: 198, 199	shtags.pl: 235, 237
routef: 175, 176	shuf: 168, 171
routel: 175, 176	shutdown: 221, 224
rpcgen: 128, 133	shutdown: 92, 94
rtacct: 175, 176	size: 145, 146
rtcwake: 230, 233	slabtop: 163, 164
rtmon: 175, 176	sleep: 168, 171
rtpr: 175, 176	sln: 128, 133
rtstat: 175, 176	soelim: 198, 199
runcon: 168, 171	sort: 168, 171
runlevel: 221, 224	sotruss: 128, 133
runlevel: 92, 94	splain: 180, 182
runtest: 120, 120	split: 168, 171
rview: 235, 237	sprof: 128, 133
rvim: 235, 237	ss: 175, 176
s2p: 180, 182	stat: 168, 171
sash64: 241, 241	stdbuf: 168, 171
sashARCS: 241, 241	strings: 145, 146
script: 230, 233	strip: 145, 146
scriptreplay: 230, 233	stty: 168, 171
scsi_id: 95, 96	su: 159, 161
sdiff: 190, 190	sulogin: 230, 233
sed: 155, 155	sum: 168, 171
seq: 168, 171	swaplabel: 230, 233
setarch: 230, 233	swapoff: 230, 233
setcap: 154, 154	swapon: 230, 233
setfacl: 152, 153	switch_root: 230, 233
setfattr: 151, 151	symlink-tree: 185, 186
setfont: 205, 206	sync: 168, 171
setkeycodes: 205, 206	sysctl: 163, 164
setleds: 205, 206	systemctl: 221, 224
setmetamode: 205, 206	systemd: 221, 224
setsid: 230, 233	systemd-analyze: 221, 224
setterm: 230, 233	systemd-ask-password: 221, 224
setvtrgb: 205, 206	systemd-cat: 221, 224
sfdisk: 230, 233	systemd-cgls: 221, 224

systemd-cgtop: 221, 224	tsort: 168, 172
systemd-coredumpctl: 221, 224	tty: 168, 172
systemd-delta: 221, 224	tune2fs: 165, 167
systemd-detect-virt: 221, 224	tzselect: 128, 134
systemd-inhibit: 221, 224	udevadm: 221, 225
systemd-machine-id-setup: 221, 224	udevadm: 95, 96
systemd-notify: 221, 224	udev: 95, 96
systemd-nspawn: 221, 224	ul: 230, 233
systemd-run: 221, 224	umount: 230, 233
systemd-stdio-bridge: 221, 224	uname: 168, 172
systemd-tmpfiles: 221, 224	uncompress: 202, 202
systemd-tty-ask-password-agent: 221, 224	unexpand: 168, 172
tabs: 157, 158	unicode_start: 205, 206
tac: 168, 171	unicode_stop: 205, 206
tail: 168, 171	uniq: 168, 172
tailf: 230, 233	unlink: 168, 172
tar: 228, 228	unlzma: 212, 213
taskset: 230, 233	unshare: 230, 233
tbl: 198, 199	unxz: 212, 213
tc: 175, 176	updatedb: 193, 193
tcsh: 118, 118	uptime: 163, 164
tcsh-version: 118, 118	usb_id: 95, 96
tcldtags: 235, 237	useradd: 159, 161
tee: 168, 171	userdel: 159, 161
telinit: 221, 224	usermod: 159, 161
telinit: 92, 94	users: 168, 172
test: 168, 171	utmpdump: 230, 233
texi2dvi: 229, 229	uuuid: 230, 233
texi2pdf: 229, 229	uuidgen: 230, 233, 230, 233
texindex: 229, 229	v4l_id: 95, 96
tfmtodit: 198, 200	vdir: 168, 172
tic: 157, 158	vi: 235, 237
timedatectl: 221, 224	view: 235, 237
timeout: 168, 171	vigr: 159, 161
tload: 163, 164	vim: 235, 237
toe: 157, 158	vim132: 235, 237
top: 163, 164	vim2html.pl: 235, 237
touch: 168, 171	vimdiff: 235, 237
tput: 157, 158	vimm: 235, 237
tr: 168, 172	vimspell.sh: 235, 237
tracpath: 204, 204	vimtutor: 235, 237
tracpath6: 204, 204	vipw: 159, 161
traceroute6: 204, 204	vmstat: 163, 164
troff: 198, 200	w: 163, 164
true: 168, 172	wall: 230, 233
truncate: 168, 172	watch: 163, 164
tset: 157, 158	wc: 168, 172

whatis: 208, 210
 whereis: 230, 233
 who: 168, 172
 whoami: 168, 172
 wipefs: 230, 234
 write: 230, 234
 write_cd_rules: 95, 96
 write_net_rules: 95, 96
 xargs: 193, 193
 xgettext: 194, 195
 xmlwf: 214, 214
 xsubpp: 180, 182
 xtrace: 128, 134
 xxd: 235, 237
 xz: 212, 213
 xzcat: 212, 213
 xzcmp: 212, 213
 xzdec: 212, 213
 xzdiff: 212, 213
 xzegrep: 212, 213
 xzfgrep: 212, 213
 xzgrep: 212, 213
 xzless: 212, 213
 xzmore: 212, 213
 yacc: 144, 144
 yes: 168, 172
 ylwrap: 185, 186
 zcat: 202, 202
 zcmp: 202, 202
 zdiff: 202, 202
 zdump: 128, 134
 zegrep: 202, 202
 zfgrep: 202, 202
 zforce: 202, 202
 zgrep: 202, 202
 zic: 128, 134
 zipdetails: 180, 182
 zless: 202, 203
 zmore: 202, 203
 znew: 202, 203
 zsoelim: 208, 210

libasprintf: 194, 195
 libatomic: 148, 150
 libattr: 151, 151
 libbfd: 145, 146
 libblkid: 230, 234
 libBrokenLocale: 128, 134
 libbz2*: 177, 178
 libc: 128, 134
 libcap: 154, 154
 libcc1: 148, 150
 libcheck.{a,so}: 66, 66
 libcidn: 128, 134
 libcilkrts: 148, 150
 libcom_err: 165, 167
 libcrypt: 128, 134
 libcursesw: 157, 158
 libdbus-1: 226, 227
 libdl: 128, 134
 libe2p: 165, 167
 libexpat: 214, 214
 libexpect-5.43: 119, 119
 libext2fs: 165, 167
 libfl: 143, 143, 143, 143
 libformw: 157, 158
 libg: 128, 134
 libgcc*: 148, 150
 libgcov: 148, 150
 libgdbm: 179, 179
 libgdbm_compat: 179, 179
 libgettextlib: 194, 195
 libgettextpo: 194, 195
 libgettextsrc: 194, 195
 libgmp: 137, 137
 libgmpxx: 137, 138
 libgomp: 148, 150
 libhistory: 183, 183
 libiberty: 148, 150
 libieee: 128, 134
 libisl: 141, 141
 libitm*: 148, 150
 liblsan: 148, 150
 libltdl: 174, 174
 liblto_plugin: 148, 150
 liblzma: 212, 213
 libm: 128, 134
 libmagic: 191, 191
 libman: 208, 210

Libraries

ld.so: 128, 134
 libacl: 152, 153
 libanl: 128, 134
 libasan: 148, 150

libmandb: 208, 210
 libmcheck: 128, 134
 libmemusage: 128, 134
 libmenuw: 157, 158
 libmount: 230, 234
 libmpc: 140, 140
 libmpfr: 139, 139
 libncursesw: 157, 158
 libnsl: 128, 134
 libnss: 128, 134
 libnss_myhostname: 221, 225
 libopcodes: 145, 147
 libpanelw: 157, 158
 libpcprofile: 128, 134
 libpipeline: 207, 207
 libprocps: 163, 164
 libpthread: 128, 134
 libquadmath*: 148, 150
 libquota: 165, 167
 libreadline: 183, 183
 libresolv: 128, 134
 librpcsvc: 128, 134
 librt: 128, 134
 libSegFault: 128, 134
 libss: 165, 167
 libssp*: 148, 150
 libstdbuf: 168, 172
 libstdc++: 148, 150
 libsupc++: 148, 150
 libsystemd: 221, 225
 libtcl-version.so: 118, 118
 libtclstub-version.a: 118, 118
 libthread_db: 128, 134
 libtsan: 148, 150
 libubsan: 148, 150
 libudev: 221, 225
 libudev: 95, 96
 libutil: 128, 134
 libuuid: 230, 234
 libvtv: 148, 150
 liby.a: 144, 144
 libz: 142, 142
 preloadable_libintl.so: 194, 195

Scripts

checkfs: 87, 87
 cleanfs: 87, 87

clock
 configuring: 244
 console
 configuring: 245
 udev: 87, 88
 functions: 87, 87
 halt: 87, 87
 hostname
 configuring: 256
 ifdown: 260, 260
 ifup: 260, 260
 ipv4-static: 260, 260
 localnet: 87, 87
 /etc/hosts: 256
 mountfs: 87, 87
 mountkernfs: 87, 87
 network
 /etc/hosts: 256
 configuring: 257
 rc: 87, 88
 reboot: 87, 88
 sendsignals: 87, 88
 setclock: 87, 88
 swap: 87, 88

Others

/boot/config-[linux-version]: 266, 269
 /boot/System.map-[linux-version]: 266, 269
 /dev/*
 boot: 103
 /etc/clfs-release: 272
 /etc/fstab
 boot: 105
 system config: 255
 /etc/group
 boot: 103
 chroot: 113
 /etc/hosts: 256
 /etc/inittab: 92
 /etc/inputrc: 253
 /etc/ld.so.conf: 132
 /etc/locale.conf: 252
 /etc/localtime: 130
 /etc/login.defs: 160
 /etc/nsswitch.conf: 130
 /etc/os-release: 223
 /etc/passwd

boot: 103
chroot: 113
/etc/profile: 251
/etc/protocols: 173
/etc/resolv.conf: 257
/etc/services: 173
/etc/udev: 95, 96
/etc/vimrc: 236
/lib/udev: 95, 96
/usr/include/{asm,linux}/*.h: 126, 126
dhcpcd: 263
man pages: 127, 127