

Cross-Compiled Linux From Scratch

Version 2.1.0-PowerPC

Cross-Compiled Linux From Scratch: Version 2.1.0-PowerPC

Copyright © 2005–2013 Joe Ciccone, Jim Gifford & Ryan Oliver

Based on LFS, Copyright © 1999–2013 Gerard Beekmans

Copyright © 2005–2013, 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 © 1999–2013, 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. Master Changelog	4
1.4. Changelog for PowerPC	8
1.5. Resources	8
1.6. Help	9
II. Preparing for the Build	12
2. Preparing a New Partition	13
2.1. Introduction	13
2.2. Creating a New Partition	13
2.3. Creating a File System on the Partition	13
2.4. Mounting the New Partition	14
3. Packages and Patches	16
3.1. Introduction	16
3.2. All Packages	16
3.3. Additional Packages for PowerPC	23
3.4. Needed Patches	23
3.5. Additional Patches for PowerPC	25
4. Final Preparations	26
4.1. About $\{\text{CLFS}\}$	26
4.2. Creating the $\{\text{CLFS}\}/\text{tools}$ Directory	26
4.3. Creating the $\{\text{CLFS}\}/\text{cross-tools}$ Directory	27
4.4. Adding the CLFS User	27
4.5. Setting Up the Environment	28
4.6. About the Test Suites	29
III. Make the Cross-Compile Tools	30
5. Constructing Cross-Compile Tools	31
5.1. Introduction	31
5.2. Build CFLAGS	31
5.3. Build Variables	32
5.4. Bc-1.06.95	33
5.5. Linux-Headers-3.10.14	34
5.6. File-5.15	35
5.7. M4-1.4.17	36
5.8. Ncurses-5.9	37

5.9. GMP-5.1.3	38
5.10. MPFR-3.1.2	39
5.11. MPC-1.0.1	40
5.12. ISL-0.12.1	41
5.13. CLooG-0.18.0	42
5.14. Cross Binutils-2.23.2	43
5.15. Cross GCC-4.8.1 - Static	45
5.16. EGLIBC-2.18	48
5.17. Cross GCC-4.8.1 - 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-5.1.3	54
6.4. MPFR-3.1.2	55
6.5. MPC-1.0.1	56
6.6. ISL-0.12.1	57
6.7. CLooG 0.18.0	58
6.8. Zlib-1.2.8	59
6.9. Binutils-2.23.2	60
6.10. GCC-4.8.1	61
6.11. Ncurses-5.9	63
6.12. Bash-4.2	64
6.13. Bison-3.0	66
6.14. Bzip2-1.0.6	67
6.15. Coreutils-8.21	68
6.16. Diffutils-3.3	69
6.17. Findutils-4.4.2	70
6.18. File-5.15	71
6.19. Flex-2.5.37	72
6.20. Gawk-4.1.0	73
6.21. Gettext-0.18.3.1	74
6.22. Grep-2.14	75
6.23. Gzip-1.6	76
6.24. M4-1.4.17	77
6.25. Make-3.82	78
6.26. Patch-2.7.1	79
6.27. Sed-4.2.2	80
6.28. Tar-1.26	81
6.29. Texinfo-4.13a	82
6.30. Vim-7.4	83
6.31. XZ Utils-5.0.5	85
6.32. To Boot or to Chroot?	86
7. If You Are Going to Boot	87
7.1. Introduction	87
7.2. Creating Directories	87
7.3. Creating Essential Symlinks	88

7.4. Util-linux-2.23.2	89
7.5. Shadow-4.1.5.1	90
7.6. E2fsprogs-1.42.8	91
7.7. Sysvinit-2.88dsf	92
7.8. Kmod-15	94
7.9. Eudev-1.3	95
7.10. Creating the passwd, group, and log Files	96
7.11. Linux-3.10.14	99
7.12. Hfsutils-3.2.6	101
7.13. Powerpc-Utills_1.1.3	102
7.14. Yaboot-1.3.17	103
7.15. Setting Up the Environment	104
7.16. Creating the /etc/fstab File	104
7.17. Bootscripts for CLFS 2.1-pre1	106
7.18. Populating /dev	107
7.19. Changing Ownership	107
7.20. Making the Temporary System Bootable	107
7.21. What to do next	108
8. If You Are Going to Chroot	109
8.1. Introduction	109
8.2. Util-linux-2.23.2	110
8.3. Mounting Virtual Kernel File Systems	111
8.4. Before Entering the Chroot Environment	112
8.5. Entering the Chroot Environment	113
8.6. Changing Ownership	114
8.7. Creating Directories	114
8.8. Creating Essential Symlinks	115
8.9. Creating the passwd, group, and log Files	115
8.10. Mounting Kernel Filesystems	117
V. Building the CLFS System	118
9. Constructing Testsuite Tools	119
9.1. Introduction	119
9.2. Tcl-8.6.1	120
9.3. Expect-5.45	121
9.4. DejaGNU-1.5.1	122
9.5. Check-0.9.10	123
10. Installing Basic System Software	124
10.1. Introduction	124
10.2. Package Management	124
10.3. About Test Suites, Again	127
10.4. Temporary Perl-5.18.1	128
10.5. Linux-Headers-3.10.14	129
10.6. Man-pages-3.54	130
10.7. EGLIBC-2.18	131
10.8. Adjusting the Toolchain	138
10.9. GMP-5.1.3	139
10.10. MPFR-3.1.2	140

10.11. MPC-1.0.1	141
10.12. ISL-0.12.1	142
10.13. CLooG-0.18.0	143
10.14. Zlib-1.2.8	144
10.15. Binutils-2.23.2	145
10.16. GCC-4.8.1	148
10.17. Sed-4.2.2	151
10.18. Ncurses-5.9	152
10.19. Pkg-config-lite-0.28-1	154
10.20. Util-linux-2.23.2	155
10.21. Procps-3.2.8	159
10.22. E2fsprogs-1.42.8	161
10.23. Shadow-4.1.5.1	164
10.24. Coreutils-8.21	167
10.25. Iana-Etc-2.30	172
10.26. M4-1.4.17	173
10.27. Bison-3.0	174
10.28. Libtool-2.4.2	175
10.29. Flex-2.5.37	176
10.30. IPRoute2-3.10.0	177
10.31. Perl-5.18.1	179
10.32. Readline-6.2	182
10.33. Autoconf-2.69	183
10.34. Automake-1.12.4	184
10.35. Bash-4.2	186
10.36. Bc-1.06.95	187
10.37. Bzip2-1.0.6	188
10.38. Diffutils-3.3	190
10.39. File-5.15	191
10.40. Gawk-4.1.0	192
10.41. Findutils-4.4.2	193
10.42. Gettext-0.18.3.1	195
10.43. Grep-2.14	197
10.44. Groff-1.22.2	198
10.45. Less-460	201
10.46. Gzip-1.6	202
10.47. IPutils-s20121221	203
10.48. Kbd-2.0.0	204
10.49. Make-3.82	206
10.50. XZ-Utils-5.0.5	207
10.51. Man-1.6g	209
10.52. Kmod-15	211
10.53. Patch-2.7.1	213
10.54. Psmisc-22.20	214
10.55. Libestr-0.1.5	215
10.56. Libee-0.4.1	216
10.57. Rsyslog-6.4.2	217

10.58. Sysvinit-2.88dsf	220
10.59. Tar-1.26	223
10.60. Texinfo-4.13a	224
10.61. Eudev-1.3	226
10.62. Vim-7.4	228
10.63. Hfsutils-3.2.6	231
10.64. Parted-3.1	232
10.65. Powerpc-Utills_1.1.3	233
10.66. Yaboot-1.3.17	234
10.67. About Debugging Symbols	235
10.68. Stripping	235
11. Setting Up System Bootscripts	236
11.1. Introduction	236
11.2. Bootscripts for CLFS 2.1-pre1	237
11.3. How Do These Bootscripts Work?	239
11.4. Configuring the setlock Script	240
11.5. Configuring the Linux Console	240
11.6. Device and Module Handling on a CLFS System	241
11.7. Creating custom symlinks to devices	244
11.8. The Bash Shell Startup Files	246
11.9. Setting Up Locale Information	246
11.10. Creating the /etc/inputrc File	248
12. Networking Configuration	250
12.1. Configuring the localnet Script	250
12.2. Customizing the /etc/hosts File	250
12.3. Creating the /etc/resolv.conf File	251
12.4. DHCP or Static Networking?	251
12.5. Static Networking Configuration	252
12.6. DHCPD-6.1.0	253
12.7. DHCP Networking Configuration	254
13. Making the CLFS System Bootable	255
13.1. Introduction	255
13.2. Creating the /etc/fstab File	255
13.3. Linux-3.10.14	256
13.4. Making the CLFS System Bootable	259
14. The End	262
14.1. The End	262
14.2. Download Client	262
14.3. Rebooting the System	263
14.4. What Now?	264
VI. Appendices	266
A. Acronyms and Terms	267
B. Dependencies	270
C. PowerPC Dependencies	279
D. Open Firmware and Mac issues.	280
E. Package Rationale	282
F. Open Publication License	287

Index 290

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 Alpha, 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 Utley, 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. 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.linuxhq.com/guides/LUG/guide.html>

This guide covers the usage of assorted Linux software.

- The Essential Pre-Reading Hint

http://hints.cross-lfs.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 2.23.2 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 4.8.1 are not recommended as they have not been tested)
- **Glibc-2.2.5** (Versions greater than 2.18 are not recommended as they have not been tested)
- **Grep-2.5**
- **Gzip-1.2.4**
- **Linux 2.6.32 (Built with GCC 4.1.2 or later)**
- **Make-3.80**
- **Ncurses-5.3**
- **Patch-2.5.4**
- **Sed-3.0.2**
- **Tar-1.22**
- **Texinfo-4.7**
- **XZ-Utills-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
uname -s -r
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 '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://cross-lfs.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.cross-lfs.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:

- Ryan Oliver - Build Process Developer.
- Jim Gifford - Lead Developer.
- Joe Ciccone - Lead Developer.
- Jeremy Utley - Release Manager 1.x Series.

Our CLFS Team:

- Nathan Coulson - Bootscripts.
- Matt Darcy - x86, X86_64, and Sparc builds.
- Manuel Canales Esparcia - Book XML.
- Karen McGuinness - Proofreader.
- Jonathan Norman - x86, x86_64, PowerPC & UltraSPARC.
- Jeremy Huntwork - PowerPC, x86, Sparc builds.
- Justin Knierim - Website Architect.
- Ken Moffat - PowerPC and X86_64 builds. Developer of Pure 64 Hint.
- Alexander E. Patrakov - Udev/Hotplug Integration
- Chris Staub - x86 builds. Leader of Quality Control.
- Zack Winkles - Unstable book work.
- William Harrington - x86, x86_64, PowerPC, Sparc, Mips builds.

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.cross-lfs.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.
- William Zhou - Text updates and Typos
- Theo Schneider - Testing of the Linux Headers Package

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, Mandriva, SUSE, 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`. This minimal system will be built using the cross-toolchain in `/cross-tools`.

In Installing Basic System Software, the full CLFS system is built. Depending on the system you are cross-compiling for, you will either boot the minimal temp-system on the target machine, or `chroot` into it.

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.

Some systems cannot be built by chrooting so they must be booted instead. Generally, if you building for a different arch than the host system, you must reboot because the kernel will likely not support the target machine. Booting involves installing a few additional packages that are needed for bootup, installing bootscripts, and building a minimal kernel. We also describe some alternative booting methods in Section 7.21, “What to do next”

To finish the installation, the CLFS-Bootscripts are set up in Setting Up System Bootscripts, 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. Master Changelog

This is version 2.1.0 of the Cross-Compiled Linux From Scratch book, dated October 06, 2013. If this book is more than six months old, a newer and better version is probably already available. To find out, please check one of the mirrors via <http://trac.cross-lfs.org/>.

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

Changelog Entries:

- 06 October 2013
 - [William Harrington] - Fix shm umount during end section reboot.
- 24 September 2013
 - [William Harrington] - Update File to 5.15.
- 23 September 2013
 - [William Harrington] - Update M4 to 1.4.17.
 - [William Harrington] - Update EUDEV to 1.3.
 - [William Harrington] - Update DHCPCD to 6.1.0.
 - [William Harrington] - Update TCL to 8.6.1.
 - [William Harrington] - Update Man-pages to 3.54.
- 19 September 2013
 - [William Harrington] - Remove pt_chown entries from book.
- 09 September 2013
 - [William Harrington] - Increase the stack size during the GCC testsuite.
- 08 September 2013
 - [William Harrington] - Change GID of tty to 5 and tape to 4.
- 03 September 2013
 - [William Harrington] - Fix final system EUDEV installation.
- 30 August 2013
 - [William Harrington] - Move GDB python files auto-load directory.
- 28 August 2013
 - [William Harrington] - Add GCC 4.8.1 branch update patch.
- 23 August 2013
 - [William Harrington] - Update KMOD to 15.
 - [William Harrington] - Update KBD to 2.0.0.
 - [William Harrington] - Add Check to test-suite tools.
- 22 August 2013
 - [William Harrington] - Install firmware option during boot method.
 - [William Harrington] - Add rule to Eudev installs for proper ethernet device naming.
- 21 August 2013
 - [William Harrington] - Update EGLIBC to 2.18-r23806.

- [William Harrington] - Update Linux to 3.10.9.
- 19 August 2013
 - [William Harrington] - Add missing **make mrproper** to final-system linux headers install.
 - [William Harrington] - Add M4=m4 to temp system configure commands of Bison and Flex as M4 is hardcoded to /cross-tools.
- 16 August 2013
 - [William Harrington] - Update Gettext to 0.18.3.1.
 - [William Harrington] - Update Bison to 3.0.
- 15 August 2013
 - [William Harrington] - Fix shm mount during chroot virtual kernel filesystems.
- 14 August 2013
 - [William Harrington] - Add --noclear to `inittab`.
 - [William Harrington] - Disable `fixincludes` for GCC in temp and final systems.
 - [William Harrington] - Update Perl to 5.18.1.
- 12 August 2013
 - [William Harrington] - Add `ac_cv_prog_lex_is_flex=yes` to temp-system and final-system Bison.
 - [William Harrington] - Update Vim to Vim 7.4.
 - [William Harrington] - Fix temp system GCC for HOST gmp isl and cloog libraries and headers.
- 08 August 2013
 - [William Harrington] - Migrate to Eudev.
 - [William Harrington] - Update Bootscripts for Eudev.
- 07 August 2013
 - [William Harrington] - Update Dhcpcd to 6.0.5.
- 06 August 2013
 - [William Harrington] - Update LESS to 460.
 - [William Harrington] - Update IPutils to s20121221.
 - [William Harrington] - Disable Vlock in KBD as it requires PAM.
 - [William Harrington] - Update Libestr to 0.1.5.
 - [William Harrington] - Update Rsyslog to 6.4.2.
- 05 August 2013
 - [William Harrington] - Remove `--enable-arch` from final-system Util-Linux as it is no longer used.
 - [William Harrington] - Update Bash patch to upstream version 4.2-045.
- 04 August 2013
 - [William Harrington] - Update Vim updates to last 1314 patch level.
 - [William Harrington] - Update boot method KMOD to support xz and zlib.
 - [William Harrington] - Remove unneeded config cache entry for `c_cv_func_setpgrp_void` during boot method shadow cross compilation.

- [William Harrington] - Update EGLIBC 2.17 to Revision 23679.
- [William Harrington] - Remove mpbsd configure option from final-system GMP.
- 02 August 2013
 - [William Harrington] - Add GMP, MPFR, MPC, ISL, CLoog configure options to temp system GCC configure.
- 01 August 2013
 - [William Harrington] - Update Util-Linux to 2.23.2.
 - [William Harrington] - Update Man-Pages to 3.53.
 - [William Harrington] - Update ISL to 0.12.1.
 - [William Harrington] - Add Make-3.82 fixes patch.
- 30 July 2013
 - [William Harrington] - Remove unneeded --disable-cloog-version-check from all gcc configure commands.
- 30 July 2013
 - [William Harrington] - Remove unneeded commands from linux header installation commands and adjust INSTALL_HDR_PATH variable.
- 29 July 2013
 - [William Harrington] - Enable graphite for cross-tools toolchain.
 - [William Harrington] - Remove unnecessary sed for headers and libraries for graphite.
- 27 July 2013
 - [William Harrington] - Add MD5SUMS and SHA1SUMS to packages introduction.
 - [William Harrington] - Add wget example to packages introduction.
- 25 July 2013
 - [William Harrington] - Add sed to cross-tools and temp-system binutils for hosts using Texinfo 5.x.
- 23 July 2013
 - [William Harrington] - Add Bc to cross-tools and final-system.
 - [William Harrington] - Update Linux kernel version to 3.10.2.
 - [William Harrington] - Update IPRoute2 to 3.10.0.
- 22 July 2013
 - [William Harrington] - Update KBD to 1.15.5.
 - [William Harrington] - Remove unneeded KBD es_po patch.
- 09 July 2013
 - [William Harrington] - Update Gettext to 0.18.3.
 - [William Harrington] - Remove unneeded config.cache command for Temp System Patch.
- 08 July 2013
 - [William Harrington] - Remove unneeded entries to cross-tools EGLIBC config.cache and change description.
 - [William Harrington] - Remove unneeded inst_varbdbir install option for current EGLIBC.
- 07 July 2013

- [William Harrington] - Update TZData to 2013d.
- 06 July 2013
 - [William Harrington] - Update Man-Pages to 3.52.
- 03 July 2013
 - [William Harrington] - Update KMOD to 14.
- 25 June 2013
 - [William Harrington] - Add ISL 0.12 to the book.
- 24 June 2013
 - [William Harrington] - Updated Gzip to 1.6.
 - [William Harrington] - Update E2fsprogs to 1.42.8.
- 07 June 2013
 - [William Harrington] - Add a sed to Tar-1.26 installs as gets() is no longer declared with Glibc-2.17.
 - [William Harrington] - Update Vim branch update patch to level 1140.
- 05 June 2013
 - [William Harrington] - Update Eglibc to 2.17.
 - [William Harrington] - Remove Eglibc fixes patch.
 - [William Harrington] - Add TZ Data to final systme Eglibc install.
- 03 June 2013
 - [William Harrington] - Update Iproute2 to 3.8.0.
 - [William Harrington] - Update GCC to 4.8.1.
 - [William Harrington] - Update CLooG to 0.18.0.
 - [William Harrington] - Remove PPL.
 - [William Harrington] - Remove -fexceptions from GMP builds.
 - [William Harrington] - Update CLooG build instructions.
- 02 June 2013
 - [William Harrington] - Add a sed for temp-system Gawk extension error.
- 31 May 2013
 - [William Harrington] - Update Coreutils to 8.21.
 - [William Harrington] - Update Bison to 2.7.1.
 - [William Harrington] - Update Util-linux to 2.23.1.
 - [William Harrington] - Expand tcl space for regular expressions required by some tests.
- 28 May 2013
 - [William Harrington] - Update File to 5.14.
- 27 May 2013
 - [William Harrington] - Update Zlib to 1.2.8.
 - [William Harrington] - Add g++ to host reqs test script.
 - [William Harrington] - Update MPFR to 3.1.2.

- [William Harrington] - Update GMP to 5.1.2.
- [William Harrington] - Update Binutils to 2.23.2.
- [William Harrington] - Update DejaGNU to 1.5.1.
- [William Harrington] - Update Diffutils to 3.3.
- [William Harrington] - Update E2fsprogs to 1.42.7.
- [William Harrington] - Update Gawk to 4.1.0.
- [William Harrington] - Update GMP to 5.1.2.
- [William Harrington] - Update Gettext to 0.18.2.1.
- [William Harrington] - Update Groff to 1.22.2.
- [William Harrington] - Update KMOD to 13.
- [William Harrington] - Update Less to 459.
- [William Harrington] - Update Linux to 3.8.13.
- [William Harrington] - Man-Pages to 3.51.
- [William Harrington] - Update Perl to 5.18.0.
- [William Harrington] - Update Pkg-Config-Lite to 0.28-1.
- [William Harrington] - Update Sed to 4.2.2.
- [William Harrington] - Update TCL to 8.6.0.
- 24 April 2013
 - [William Harrington] - Changelog restarted, see the 2.0.0 book for the old changelog.

1.4. Changelog for PowerPC

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

Changelog Entries:

- 24 April 2013
 - [William Harrington] - Changelog restarted, see the 2.0.0 book for the old changelog.

1.5. Resources

1.5.1. 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.cross-lfs.org/wiki/faq>.

1.5.2. Mailing Lists

The `cross-lfs.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.cross-lfs.org/wiki/lists>.

1.5.3. 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](http://gmane.org) search for CLFS with the following link:

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

1.5.4. 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.cross-lfs.org> and reference the pastebin URL when asking your questions.

1.5.5. 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.cross-lfs.org/wiki/mirrors> for mirrors of CLFS.

1.5.6. Contact Information

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

1.6. Help

If an issue or a question is encountered while working through this book, check the FAQ page at <http://trac.cross-lfs.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.cross-lfs.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 1.5, “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.

1.6.1. 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 2.1.0)

- The host distribution and version being used to create CLFS.
- The architecture of the host and target.
- The value of the \$CLFS_HOST, \$CLFS_TARGET, \$BUILD32, and \$BUILD64 environment variables.
- The package or section in which the problem was encountered.
- The exact error message or symptom received. See Section 1.6.3, “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.

1.6.2. 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.

1.6.3. 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. 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” and transfer it to your target machine.

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 an 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.

Open Firmware and the Mac OS's impose certain requirements on partitioning. This is discussed in Appendix E. In particular, you cannot use **fdisk**, you will need an `apple_bootstrap` partition, and that should precede any OSX partition.

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

Remember the designation of the new partition (e.g., `hda5`). 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. You will also need to know the designation of the `apple_bootstrap` partition for the `yaboot.conf` when you set this up before you run **ybin**.

2.3. Creating a File System on the Partition

Now that a blank partition has been set up, the file system can be created. The most widely-used system in the Linux world is the second extended file system (`ext2`), but with newer high-capacity hard disks, journaling file systems are becoming increasingly popular. We will create an `ext2` file system. Instructions for other file systems can be found at http://cblfs.cross-lfs.org/index.php?section=6#File_System.

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

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

Replace `[xxx]` with the name of the CLFS partition (`hda5` in our previous example).



Note

Some host distributions use custom features in their filesystem creation tools (E2fsprogs). This can cause problems when booting into your new 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`; `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:

```
cd /tmp
tar xjf /path/to/sources/e2fsprogs-1.42.8.tar.bz2
cd e2fsprogs-1.42.8
mkdir build
cd build
../configure
make #note that we intentionally don't 'make install' here!
./misc/mke2fs /dev/[xxx]
cd /tmp
rm -rf e2fsprogs-1.42.8
```

If a swap partition was created, it will need to be initialized for use by issuing the command below. If you are using an existing swap partition, there is no need to format it.

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

Replace `[yyy]` with the name of the swap partition.

2.4. Mounting the New Partition

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

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

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

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

```
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 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 the **mount** command without any parameters 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://cross-lfs.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 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
md5sum -c SHA1SUMS
popd
```

3.2. All Packages

Download or otherwise obtain the following packages:

- **Autoconf (2.69) - 1,188 KB:**

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

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

MD5 sum: 50f97f4159805e374639a73e2636f22e

- **Automake (1.12.4) - 1,356 KB:**

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

Download: <http://ftp.gnu.org/gnu/automake/automake-1.12.4.tar.xz>

MD5 sum: 7395a0420ecb5c9bc43e5fcf4824df36

- **Bash (4.2) - 6,848 KB:**

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

Download: <http://ftp.gnu.org/gnu/bash/bash-4.2.tar.gz>

MD5 sum: 3fb927c7c33022f1c327f14a81c0d4b0

- **Bc (1.06.95) - 284 KB:**

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

Download: <http://alpha.gnu.org/gnu/bc/bc-1.06.95.tar.bz2>

MD5 sum: 5126a721b73f97d715bb72c13c889035

- **Binutils (2.23.2) - 20,940 KB:**

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

Download: <http://ftp.gnu.org/gnu/binutils/binutils-2.23.2.tar.bz2>

MD5 sum: 4f8fa651e35ef262edc01d60fb45702e

- **Bison (3.0) - 1,914 KB:**

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

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

MD5 sum: a2624994561aa69f056c904c1ccb2880

- **Bootscripts for CLFS (2.1-pre1) - 41 KB:**

Download: <http://cross-lfs.org/files/packages/git/bootscripts-cross-lfs-2.1-pre1.tar.xz>

MD5 sum: f474bf2efff744548a69d9049bad973f

- **Bzip2 (1.0.6) - 764 KB:**

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

Download: <http://www.bzip.org/1.0.6/bzip2-1.0.6.tar.gz>

MD5 sum: 00b516f4704d4a7cb50a1d97e6e8e15b

- **Check (0.9.10) - 650 KB:**

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

Download: <http://sourceforge.net/projects/check/files/check/0.9.10/check-0.9.10.tar.gz>

MD5 sum: 6d10a8efb9a683467b92b3bce97aeb30

- **Cloog (0.18.0) - 3,688 KB:**

Home page: <http://cloog.org>

Download: <http://www.bastoul.net/cloog/pages/download/cloog-0.18.0.tar.gz>

MD5 sum: be78a47bd82523250eb3e91646db5b3d

- **Coreutils (8.21) - 5,236 KB:**

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

Download: <http://ftp.gnu.org/gnu/coreutils/coreutils-8.21.tar.xz>

MD5 sum: 065ba41828644eca5dd8163446de5d64

- **DejaGNU (1.5.1) - 568 KB:**

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

Download: <http://ftp.gnu.org/gnu/dejagnu/dejagnu-1.5.1.tar.gz>

MD5 sum: 8386e04e362345f50ad169f052f4c4ab

- **DHCPD (6.1.0) - 114 KB KB:**

Home page: <http://roy.marples.name/projects/dhcpd>

Download: <http://roy.marples.name/downloads/dhcpd/dhcpd-6.1.0.tar.bz2>

MD5 sum: 6070040c57492925af9ac6aed980de2a

- **Diffutils (3.3) - 1,172 KB:**

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

Download: <http://ftp.gnu.org/gnu/diffutils/diffutils-3.3.tar.xz>

MD5 sum: 99180208ec2a82ce71f55b0d7389f1b3

- **EGLIBC (2.18) - 11,943 KB:**

Home page: <http://www.eglibc.org/home>

Download: <http://cross-lfs.org/files/eglibc-2.18-r24148.tar.xz>

MD5 sum: 8b3dc01f6ee5f1654b98213e8d4721a4

- **E2fsprogs (1.42.8) - 4,496 KB:**

Home page: <http://e2fsprogs.sourceforge.net>

Download: <http://www.kernel.org/pub/linux/kernel/people/tytso/e2fsprogs/v1.42.8/e2fsprogs-1.42.8.tar.xz>

MD5 sum: 57f20ba5e4cac8ce082065a61aa3f3bc

- **Eudev (1.3) - 1,679 KB:**

Home page: <http://www.gentoo.org/proj/en/eudev/>

Download: <ftp://mirror.ovh.net/gentoo-distfiles/distfiles/eudev-1.3.tar.gz>

MD5 sum: 164df78f6f0093578a20bdd00335845f

- **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 (5.15) - 656 KB:**

Home page: <http://www.darwinsys.com/file>

Download: <ftp://ftp.astron.com/pub/file/file-5.15.tar.gz>

MD5 sum: 3f99565532f548d7540912c4642d1ede



Note

File (5.15) 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://cross-lfs.org/files/packages/git/>.

- **Findutils (4.4.2) - 2,100 KB:**

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

Download: <http://ftp.gnu.org/gnu/findutils/findutils-4.4.2.tar.gz>

MD5 sum: 351cc4adb07d54877fa15f75fb77d39f

- **Flex (2.5.37) - 1,276 KB:**

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

Download: <http://downloads.sourceforge.net/flex/flex-2.5.37.tar.bz2>

MD5 sum: c75940e1fc25108f2a7b3ef42abdae06

• **Gawk (4.1.0) - 2,004 KB:**

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

Download: <http://ftp.gnu.org/gnu/gawk/gawk-4.1.0.tar.xz>

MD5 sum: b18992ff8faf3217dab55d2d0aa7d707

• **GCC (4.8.1) - 84,720 KB:**

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

Download: <ftp://gcc.gnu.org/pub/gcc/releases/gcc-4.8.1/gcc-4.8.1.tar.bz2>

MD5 sum: 3b2386c114cd74185aa3754b58a79304

• **Gettext (0.18.3.1) - 16,342 KB:**

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

Download: <http://ftp.gnu.org/gnu/gettext/gettext-0.18.3.1.tar.gz>

MD5 sum: 3fc808f7d25487fc72b5759df7419e02

• **GMP (5.1.3) - 1,819 KB:**

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

Download: <http://ftp.gnu.org/gnu/gmp/gmp-5.1.3.tar.xz>

MD5 sum: e5fe367801ff067b923d1e6a126448aa

• **Grep (2.14) - 1,168 KB:**

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

Download: <http://ftp.gnu.org/gnu/grep/grep-2.14.tar.xz>

MD5 sum: d4a3f03849d1e17ce56ab76aa5a24cab

• **Groff (1.22.2) - 3,928 KB:**

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

Download: <http://ftp.gnu.org/gnu/groff/groff-1.22.2.tar.gz>

MD5 sum: 9f4cd592a5efc7e36481d8d8d8af6d16

• **Gzip (1.6) - 812 KB:**

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

Download: <http://ftp.gnu.org/gnu/gzip/gzip-1.6.tar.xz>

MD5 sum: da981f86677d58a106496e68de6f8995

• **Iana-Etc (2.30) - 204 KB:**

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

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

MD5 sum: 3ba3afb1d1b261383d247f46cb135ee8

• **IPRoute2 (3.10.0) - 412 KB:**

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

Download: <http://www.kernel.org/pub/linux/utils/net/iproute2/iproute2-3.10.0.tar.xz>

MD5 sum: 45fb5427fc723a0001c72b92c931ba02

• **IPutils (s20121221) - 155 KB:**

Home page: <http://www.linuxfoundation.org/en/Net:Iputils>

Download: <http://www.skbuff.net/iputils/iputils-s20121221.tar.bz2>

MD5 sum: 6072aef64205720dd1893b375e184171

• **ISL (0.12.1) - 1,161 KB:**

Home page: <http://garage.kotnet.org/~skimo/isl/>

Download: <http://isl.gforge.inria.fr/isl-0.12.1.tar.lzma>

MD5 sum: d7a723a508056b9dc5a25c5ca7d1d74f

• **Kbd (2.0.0) - 2,007 KB:**

Download: <ftp://devel.altlinux.org/legion/kbd/kbd-2.0.0.tar.gz>

MD5 sum: 5ba259a0b2464196f6488a72070a3d60

• **Kmod (15) - 1,454 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-15.tar.xz>

MD5 sum: d03372179ed2cfa0c52b6672cf438901

• **Less (460) - 311 KB:**

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

Download: <http://www.greenwoodsoftware.com/less/less-460.tar.gz>

MD5 sum: c3b603140aed2beb6091fdbbc27f80ff0

• **Libee (0.4.1) - 352 KB:**

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

Download: <http://www.libee.org/download/files/download/libee-0.4.1.tar.gz>

MD5 sum: 7bbf4160876c12db6193c06e2badedb2

• **Libestr (0.1.5) - 326 KB:**

Home page: <http://libestr.adiscon.com/>

Download: <http://libestr.adiscon.com/files/download/libestr-0.1.5.tar.gz>

MD5 sum: f180c0cdc82883d161eba3f2e8a34eb4

• **Libtool (2.4.2) - 852 KB:**

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

Download: <http://ftp.gnu.org/gnu/libtool/libtool-2.4.2.tar.xz>

MD5 sum: 2ec8997e0c07249eb4cbd072417d70fe

• **Linux (3.10.14) - 73,212 KB:**

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

Download: <http://www.kernel.org/pub/linux/kernel/v3.0/linux-3.10.14.tar.xz>

MD5 sum: 3cd1e4b50fb9decd63754ae80f3b2414

• **M4 (1.4.17) - 1,149 KB:**

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

Download: <http://ftp.gnu.org/gnu/m4/m4-1.4.17.tar.xz>

MD5 sum: 12a3c829301a4fd6586a57d3fcf196dc

• **Make (3.82) - 1,216 KB:**

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

Download: <http://ftp.gnu.org/gnu/make/make-3.82.tar.bz2>

MD5 sum: 1a11100f3c63fcf5753818e59d63088f

• **Man (1.6g) - 252 KB:**

Home page: <http://primates.ximian.com/~flucifredi/man>

Download: <http://primates.ximian.com/~flucifredi/man/man-1.6g.tar.gz>

MD5 sum: ba154d5796928b841c9c69f0ae376660

• **Man-pages (3.54) - 1,172 KB:**

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

Download: <http://www.kernel.org/pub/linux/docs/man-pages/man-pages-3.54.tar.xz>

MD5 sum: 382f83e670ecbe1d97fc58e03da0b026

• **MPC (1.0.1) - 616 KB:**

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

Download: <http://www.multiprecision.org/mpc/download/mpc-1.0.1.tar.gz>

MD5 sum: b32a2e1a3daa392372fbd586d1ed3679

• **MPFR (3.1.2) - 1,050 KB:**

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

Download: <http://www.mpfr.org/mpfr-3.1.2/mpfr-3.1.2.tar.xz>

MD5 sum: e3d203d188b8fe60bb6578dd3152e05c

• **Ncurses (5.9) - 2,764 KB:**

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

Download: <ftp://ftp.gnu.org/pub/gnu/ncurses/ncurses-5.9.tar.gz>

MD5 sum: 8cb9c412e5f2d96bc6f459aa8c6282a1

• **Patch (2.7.1) - 668 KB:**

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

Download: <http://ftp.gnu.org/gnu/patch/patch-2.7.1.tar.xz>

MD5 sum: e9ae5393426d3ad783a300a338c09b72

• **Perl (5.18.1) - 14,060 KB:**

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

Download: <http://www.cpan.org/src/5.0/perl-5.18.1.tar.bz2>

MD5 sum: 4ec1a3f3824674552e749ae420c5e68c

• **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 (3.2.8) - 280 KB:**

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

Download: <http://procps.sourceforge.net/procps-3.2.8.tar.gz>

MD5 sum: 9532714b6846013ca9898984ba4cd7e0

• **Psmisc (22.20) - 428 KB:**

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

Download: <http://downloads.sourceforge.net/psmisc/psmisc-22.20.tar.gz>

MD5 sum: a25fc99a6dc7fa7ae6e4549be80b401f

• **Readline (6.2) - 2,228 KB:**

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

Download: <http://ftp.gnu.org/gnu/readline/readline-6.2.tar.gz>

MD5 sum: 67948acb2ca081f23359d0256e9a271c

• **Rsyslog (6.4.2) - 2,519 KB:**

Home page: <http://www.rsyslog.com/>

Download: <http://www.rsyslog.com/files/download/rsyslog/rsyslog-6.4.2.tar.gz>

MD5 sum: 7de0124ec7d67ce2bfda0009ab1263ee

• **Sed (4.2.2) - 1,036 KB:**

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

Download: <http://ftp.gnu.org/gnu/sed/sed-4.2.2.tar.bz2>

MD5 sum: 7ffe1c7cdc3233e1e0c4b502df253974

• **Shadow (4.1.5.1) - 2,144 KB:**

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

Download: <http://pkg-shadow.alioth.debian.org/releases/shadow-4.1.5.1.tar.bz2>

MD5 sum: a00449aa439c69287b6d472191dc2247

• **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: 6eda8a97b86e0a6f59dabbbf25202aa6f

• **Tar (1.26) - 2,288 KB:**

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

Download: <http://ftp.gnu.org/gnu/tar/tar-1.26.tar.bz2>

MD5 sum: 2cee42a2ff4f1cd4f9298eeeb2264519

• **Tcl (8.6.1) - 8,756 KB:**

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

Download: <http://downloads.sourceforge.net/tcl/tcl8.6.1-src.tar.gz>

MD5 sum: aae4b701ee527c6e4e1a6f9c7399882e

• **Texinfo (4.13a) - 2,688 KB:**

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

Download: <http://ftp.gnu.org/gnu/texinfo/texinfo-4.13a.tar.gz>

MD5 sum: 71ba711519209b5fb583fed2b3d86fcb

• **Time Zone Data (2013g) - 227 KB:**

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

Download: <http://www.iana.org/time-zones/repository/releases/tzdata2013g.tar.gz>

MD5 sum: 76dbc3b5a81913fc0d824376c44a5d15

• **Util-linux (2.23.2) - 3,383 KB:**

Home page: <http://userweb.kernel.org/~kzak/util-linux/>

Download: <http://www.kernel.org/pub/linux/utils/util-linux/v2.23/util-linux-2.23.2.tar.xz>

MD5 sum: b39fde897334a4858bb2098edcce5b3f

• **Vim (7.4) - 9,843 KB:**

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

Download: <ftp://ftp.vim.org/pub/vim/unix/vim-7.4.tar.bz2>

MD5 sum: 607e135c559be642f210094ad023dc65

- **XZ Utils (5.0.5) - 908 KB:**

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

Download: <http://tukaani.org/xz/xz-5.0.5.tar.xz>

MD5 sum: aa17280f4521dbeebed0fbd11cd7fa30

- **Zlib (1.2.8) - 440 KB:**

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

Download: <http://zlib.net/zlib-1.2.8.tar.xz>

MD5 sum: 28f1205d8dd2001f26fec1e8c2cebe37



Note

Zlib (1.2.8) 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://cross-lfs.org/files/packages/git/>.

Total size of these packages: about 312 MB

3.3. Additional Packages for PowerPC

- **Hfsutils (3.2.6) - 204 KB:**

Home page: <http://www.mars.org/home/rob/proj/hfs>

Download: <ftp://ftp.mars.org/pub/hfs/hfsutils-3.2.6.tar.gz>

MD5 sum: fa572afd6da969e25c1455f728750ec4

- **Parted (3.1) - 1,492 KB:**

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

Download: <http://ftp.gnu.org/gnu/parted/parted-3.1.tar.xz>

MD5 sum: 5d89d64d94bcfeafa9ce8f59f4b81bdcb

- **Powerpc-utils (1.1.3) - 28 KB:**

Home page: <http://packages.qa.debian.org/p/powerpc-utils.html>

Download: http://ftp.debian.org/debian/pool/main/p/powerpc-utils/powerpc-utils_1.1.3.orig.tar.gz

MD5 sum: d879b109bb8f0d726304b60b147bff13

- **Yaboot (1.3.17) - 220 KB:**

Home page: <http://yaboot.ozlabs.org>

Download: <http://yaboot.ozlabs.org/releases/yaboot-1.3.17.tar.gz>

MD5 sum: f599f52d1887a86fd798252d2946f635

Total size of these packages: about 1,944 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:

- **Bash Branch Update Patch - 58 KB:**

Download: http://patches.cross-lfs.org/2.1.0/bash-4.2-branch_update-7.patch

MD5 sum: 4dfb1ce9b5d0040eae06e66157ab213a

- **Coreutils Uname Patch - 4.8 KB:**

Download: <http://patches.cross-lfs.org/2.1.0/coreutils-8.21-uname-1.patch>

MD5 sum: 5d3a1f7196c9c07033bbd2853885fda2

- **GCC Branch Update Patch - 7,715 KB:**

Download: http://patches.cross-lfs.org/2.1.0/gcc-4.8.1-branch_update-3.patch

MD5 sum: 743b954ce42dd6193376e43ea84d7c10

- **Iana-Etc Get Fix Patch - 1.1 KB:**

Download: http://patches.cross-lfs.org/2.1.0/iana-etc-2.30-get_fix-1.patch

MD5 sum: 73aee2dc34cf4d990cc22fe323d89f27

- **Iana-Etc Protocol and Port Numbers Update - 3,760 KB:**

Download: http://patches.cross-lfs.org/2.1.0/iana-etc-2.30-numbers_update-20120610-2.patch

MD5 sum: 826fb780d13caafb7cb99b9c346f2102

- **IPUtils Fixes Patch - 153 KB:**

Download: <http://patches.cross-lfs.org/2.1.0/iputils-s20121221-fixes-1.patch>

MD5 sum: a2e77de7fd1fc4417bce0af3e6ffdfcb

- **Make fixes patch - 9,301 KB:**

Download: <http://patches.cross-lfs.org/2.1.0/make-3.82-fixes-1.patch>

MD5 sum: bca6c0167780f427f527e976d597b505

- **Man i18n Patch - 11 KB:**

Download: <http://patches.cross-lfs.org/2.1.0/man-1.6g-i18n-1.patch>

MD5 sum: a5aba0cb5a95a7945db8c882334b7dab

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

Download: http://patches.cross-lfs.org/2.1.0/ncurses-5.9-bash_fix-1.patch

MD5 sum: c6f7f2ab0ebaf7721eb266641352db

- **Ncurses Branch Update Patch - 2,492 KB:**

Download: http://patches.cross-lfs.org/2.1.0/ncurses-5.9-branch_update-4.patch

MD5 sum: c2b2dc2d31b02c218359e6218f12a72c

- **Perl Libc Patch - 1.603 KB:**

Download: <http://patches.cross-lfs.org/2.1.0/perl-5.18.1-libc-1.patch>

MD5 sum: 63eda1cc319206788ea93c58f395417c

- **Procps Fix HZ Errors Patch - 2.4 KB:**

Download: http://patches.cross-lfs.org/2.1.0/procps-3.2.8-fix_HZ_errors-1.patch

MD5 sum: 2ea4c8e9a2c2a5a291ec63c92d7c6e3b

- **Procps ps cgroup Patch - 3.1 KB:**

Download: http://patches.cross-lfs.org/2.1.0/procps-3.2.8-ps_cgroup-1.patch

MD5 sum: 3c478ef88fad23353e332b1b850ec630

- **Readline Branch Update - 4.9 KB:**

Download: http://patches.cross-lfs.org/2.1.0/readline-6.2-branch_update-3.patch

MD5 sum: af788f5b1cfc5db9efc9e0fa0268a574

- **Tar Man Page Patch - 74 KB:**

Download: <http://patches.cross-lfs.org/2.1.0/tar-1.26-man-1.patch>

MD5 sum: 074783d41f18c5c62a7cfc77e2678693

- **Texinfo New Compressors Patch - 3.0 KB:**

Download: http://patches.cross-lfs.org/2.1.0/texinfo-4.13a-new_compressors-1.patch

MD5 sum: 4ae2d3c132e21cb83b825bc691056d07

- **Vim Branch Update Patch - 460 KB:**

Download: http://patches.cross-lfs.org/2.1.0/vim-7.4-branch_update-1.patch

MD5 sum: b5fdb7f4e4cc27932a9183c8e289029d

Total size of these patches: about 23 MB

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.cross-lfs.org/2.1.0/> and acquire any additional patches to suit the system needs.

3.5. Additional Patches for PowerPC

- **GCC Specs Patch - 20 KB:**

Download: <http://patches.cross-lfs.org/2.1.0/gcc-4.8.1-specs-1.patch>

MD5 sum: 14aa064a113f2cae0f877039bb4a6357

- **HFS Utils Fixes Patch - 1.1 KB:**

Download: <http://patches.cross-lfs.org/2.1.0/hfsutils-3.2.6-fixes-1.patch>

MD5 sum: 8519f11aada2f393609d529621a9f1b1

- **Powerpc-utils Fixes Patch - 22 KB:**

Download: http://patches.cross-lfs.org/2.1.0/powerpc-utils_1.1.3-fixes-2.patch

MD5 sum: d2776b1a4977c5711037b8f1402f792a

- **Yaboot Ofpath_Path_Prefix Patch - .830 KB:**

Download: http://patches.cross-lfs.org/2.1.0/yaboot-1.3.17-ofpath_path_prefix-1.patch

MD5 sum: 3faf70e0cb4e4f62a1e8815c3452ab38

- **Yaboot Stubfuncs Patch - 4.2 KB:**

Download: <http://patches.cross-lfs.org/2.1.0/yaboot-1.3.17-stubfuncs-1.patch>

MD5 sum: b5cc91f9904383c24848040bfe6f11ae

Total size of these patches: about 48.13 KB

Chapter 4. Final Preparations

4.1. About `${CLFS}`

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

```
echo ${CLFS}
```

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

```
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.

If you haven't created the `${CLFS}` directory, do so at this time by issuing the following commands:

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

Do not forget to check that `${CLFS}` is set whenever you leave and reenter the current working environment (as when doing a “`su`” to `root` or another user).

4.2. Creating the `${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 /
```

The symlink isn't technically necessary (though the book's instructions do assume its existence), but is there mainly for consistency (because `/tools` is also symlinked to `{CLFS}/tools`) and to simplify the installation of the cross-compile 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 user `clfs` to group `clfs`.

`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
```

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, give user `clfs` ownership of this directory:

```
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
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. If the host system uses a version of `Glibc` older than `2.2.4`, having `LC_ALL` set to something other than `"POSIX"` or `"C"` (during this chapter) may cause issues if you exit the `chroot` environment and wish to return later. Setting `LC_ALL` to `"POSIX"` or `"C"` (the two are equivalent) ensures that everything will work as expected in the `chroot` environment.

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`.

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

```
source ~/.bash_profile
```

4.6. About the Test Suites

Most packages provide a test suite. Running the test suite for a newly built package is a good idea because it can provide a “sanity check” indicating that everything compiled correctly. A test suite that passes its set of checks usually proves that the package is functioning as the developer intended. It does not, however, guarantee that the package is totally bug free.

It is not possible to run testsuites when cross-compiling, so package installation instructions do not explain how to run testsuites 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. Build CFLAGS

CFLAGS and CXXFLAGS must not be set during the building of cross-tools.

To disable CFLAGS and CXXFLAGS use the following commands:

```
unset CFLAGS
unset CXXFLAGS
```

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

```
echo unset CFLAGS >> ~/.bashrc
echo unset CXXFLAGS >> ~/.bashrc
```

5.3. 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 to add "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:

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

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
```

5.4. Bc-1.06.95

The Bc package contains an arbitrary precision numeric processing language.

5.4.1. Installation of Bc

Prepare Bc 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 10.36.2, “Contents of Bc.”

5.5. Linux-Headers-3.10.14

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

5.5.1. Installation of Linux-Headers

For this step you will need the kernel tarball.

Install the kernel header files:

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

The meaning of the make commands:

make mrproper

Ensures that the kernel source dir is clean.

make ARCH=powerpc headers_check

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

make ARCH=powerpc 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.”

5.6. File-5.15

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

5.6.1. Installation of File

Prepare File for compilation:

```
./configure --prefix=/cross-tools --disable-static
```

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.

--disable-static

This tells the File 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 10.39.2, “Contents of File.”

5.7. M4-1.4.17

The M4 package contains a macro processor.

5.7.1. Installation of M4

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 10.26.2, “Contents of M4.”

5.8. Ncurses-5.9

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

5.8.1. Installation of Ncurses

The following patch fixes an issue with some Bash versions:

```
patch -Np1 -i ../ncurses-5.9-bash_fix-1.patch
```

Prepare Ncurses for compilation:

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

The meaning of the new configure options:

--without-debug

Tells Ncurses to build without debugging information.

--without-shared

This prevents Ncurses from building its shared libraries, which are not needed at this time.

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 10.18.2, “Contents of Ncurses.”

5.9. GMP-5.1.3

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

5.9.1. Installation of GMP



Note

If you are building with a host which has 32-bit user-space with a 64-bit capable CPU, cross-tools GMP will attempt to link with 64-bit libraries. Add the following variable during **configure** to force GMP's ABI:
`./configure ABI=32`

Prepare GMP for compilation:

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

The meaning of the new configure options:

`--enable-cxx`

This tells GMP to enable C++ support.

Compile the package:

```
make
```

Install the package:

```
make install
```

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

5.10. MPFR-3.1.2

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

5.10.1. Installation of MPFR

Prepare MPFR for compilation:

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

The meaning of the new configure options:

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

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

```
--enable-shared
```

This tells **configure** to build MPFR's shared 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 10.10.2, "Contents of MPFR."

5.11. MPC-1.0.1

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

5.11.1. 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 10.11.2, “Contents of MPC.”

5.12. ISL-0.12.1

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

5.12.1. Installation of ISL

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 10.12.2, “Contents of ISL.”

5.13. CLoog-0.18.0

CLoog is a library to generate code for scanning Z-polyhedra. In other words, it finds code that reaches each integral point of one or more parameterized polyhedra. GCC links with this library in order to enable the new loop generation code known as Graphite.

5.13.1. Installation of CLoog

Prepare CLoog for compilation:

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

5.14. Cross Binutils-2.23.2

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

5.14.1. 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.

Apply the following sed for hosts using Texinfo-5.x:

```
sed -i -e 's/@colophon/@@colophon/' \
      -e 's/doc@cygnus.com/doc@@cygnus.com/' bfd/doc/bfd.texinfo
```

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-2.23.2/configure \
  --prefix=/cross-tools --host=${CLFS_HOST} --target=${CLFS_TARGET} \
  --with-sysroot=${CLFS} --with-lib-path=/tools/lib --disable-nls \
  --disable-static --disable-multilib
```

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 that 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-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.

Compile the package:

```
make configure-host
make
```

The meaning of the make options:

configure-host

This checks the host environment and makes sure all the necessary tools are available to compile Binutils.

Install the package:

```
make install
```

Copy `libiberty.h` to `/tools/include` directory:

```
cp -v ../binutils-2.23.2/include/libiberty.h /tools/include
```

Details on this package are located in Section 10.15.2, “Contents of Binutils.”

5.15. Cross GCC-4.8.1 - Static

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

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

The following patch contains a number of updates to the 4.8.1 branch by the GCC developers:

```
patch -Np1 -i ../gcc-4.8.1-branch_update-3.patch
```

Make a couple of essential adjustments to the `specs` file to ensure GCC uses our build environment:

```
patch -Np1 -i ../gcc-4.8.1-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"
echo -en '\n#undef STANDARD_STARTFILE_PREFIX_2\n#define STANDARD_STARTFILE_PREFIX_2 "/tools"
```

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-4.8.1/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-nls \
  --disable-shared --with-mpfr=/cross-tools --with-gmp=/cross-tools \
  --with-isl=/cross-tools --with-cloog=/cross-tools --with-mpc=/cross-tools \
  --without-headers --with-newlib --disable-decimal-float --disable-libgomp \
  --disable-libmudflap --disable-libssp --disable-threads --disable-multilib \
  --disable-libatomic --disable-libitm --disable-lsanitizer \
  --disable-libquadmath --disable-target-libiberty --disable-target-zlib \
  --with-system-zlib --enable-cloog-backend=isl --disable-isl-version-check \
  --enable-languages=c --enable-checking=release
```

The meaning of the new configure options:

```
--with-sysroot=${CLFS}
```

Tells GCC to consider `${CLFS}` as the root file system.

```
--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-nls`

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

`--without-headers`

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

`--with-newlib`

Tells GCC that the target libc will use 'newlib'.

`--disable-decimal-float`

Disables support for the C decimal floating point extension.

`--disable-libgomp`

Disables the creation of runtime libraries used by GOMP.

`--disable-libmudflap`

Disables the creation of runtime libraries used by libmudflap.

`--disable-libssp`

Disables the use of Stack Smashing Protection for runtime libraries.

`--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.

`--disable-libatomic`

The atomic library isn't needed at this time.

`--disable-libitm`

The itm library isn't needed at this time.

`--disable-lsanitizer`

The sanitizer library isn't needed at this time.

`--disable-libquadmath`

The quadmath library isn't needed at this time.

`--enable-languages=c`

This option ensures that only the C compiler is built.

`--enable-checking=release`

This option selects the complexity of the internal consistency checks and adds error checking within the 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 10.16.2, “Contents of GCC.”

5.16. EGLIBC-2.18

The EGLIBC 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.

5.16.1. Installation of EGLIBC

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

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

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

Add the following to `config.cache` to disable `ssp` when building EGLIBC:

```
echo "libc_cv_ssp=no" > config.cache
```

Prepare EGLIBC for compilation:

```
BUILD_CC="gcc" CC="${CLFS_TARGET}-gcc" \
  AR="${CLFS_TARGET}-ar" RANLIB="${CLFS_TARGET}-ranlib" \
  ../eglibc-2.18/configure --prefix=/tools \
  --host=${CLFS_TARGET} --build=${CLFS_HOST} \
  --disable-profile --with-tls --enable-kernel=2.6.32 \
  --with-__thread --with-binutils=/cross-tools/bin \
  --with-headers=/tools/include --enable-obsolete-rpc \
  --cache-file=config.cache
```

The meaning of the new configure options:

BUILD_CC="gcc"

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

CC="\${CLFS_TARGET}-gcc"

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

AR="\${CLFS_TARGET}-ar"

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

RANLIB="\${CLFS_TARGET}-ranlib"

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

--disable-profile

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

--with-tls

This tells EGLIBC to use Thread Local Storage.

```
--enable-kernel=2.6.32
```

This tells EGLIBC to compile the library with support for 2.6.32 and later Linux kernels.

```
--with-__thread
```

This tells EGLIBC to use the `__thread` for `libc` and `libpthread` builds.

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

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

```
--with-headers=/tools/include
```

This tells EGLIBC 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.

```
--cache-file=config.cache
```

This tells EGLIBC to utilize a premade cache file.

During this stage the following warning might appear:

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

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

Compile the package:

```
make
```

Install the package:

```
make install
```

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

5.17. Cross GCC-4.8.1 - Final

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

5.17.1. Installation of GCC Cross Compiler

The following patch contains a number of updates to the 4.8.1 branch by the GCC developers:

```
patch -Np1 -i ../gcc-4.8.1-branch_update-3.patch
```

Make a couple of essential adjustments to the `specs` file to ensure GCC uses our build environment:

```
patch -Np1 -i ../gcc-4.8.1-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"
echo -en '\n#undef STANDARD_STARTFILE_PREFIX_2\n#define STANDARD_STARTFILE_PREFIX_2 "/tools"
```

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-4.8.1/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 \
  --enable-shared --disable-static --enable-languages=c,c++ \
  --enable-__cxa_atexit --enable-c99 --enable-long-long --enable-threads=posix \
  --disable-multilib --with-mpc=/cross-tools --with-mpfr=/cross-tools \
  --with-gmp=/cross-tools --with-cloog=/cross-tools \
  --enable-cloog-backend=isl --with-isl=/cross-tools \
  --disable-isl-version-check --with-system-zlib --enable-checking=release \
  --enable-libstdcxx-time
```

The meaning of the new configure options:

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

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

`--enable-__cxa_atexit`

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

`--enable-c99`

Enable C99 support for C programs.

`--enable-long-long`

Enables long long support in the compiler.


```
--enable-threads=posix
```

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

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 10.16.2, “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 one below. It is safe to ignore these messages.

```
configure: WARNING: If you wanted to set the --build type, don't use --host.
    If a cross compiler is detected then cross compile mode will be used.
```

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-5.1.3

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

6.3.1. Installation of GMP

Prepare GMP for compilation:

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

Compile the package:

```
make
```

Install the package:

```
make install
```

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

6.4. MPFR-3.1.2

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

6.4.1. Installation of MPFR

Prepare MPFR for compilation:

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

Compile the package:

```
make
```

Install the package:

```
make install
```

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

6.5. MPC-1.0.1

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 10.11.2, “Contents of MPC.”

6.6. ISL-0.12.1

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} \  
  --with-gmp-prefix=/tools
```

Compile the package:

```
make
```

Install the package:

```
make install
```

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

6.7. CLooG 0.18.0

CLooG is a library to generate code for scanning Z-polyhedra. In other words, it finds code that reaches each integral point of one or more parameterized polyhedra. GCC links with this library in order to enable the new loop generation code known as Graphite.

6.7.1. Installation of CLooG

Prepare CLooG for compilation:

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

Compile the package:

```
make
```

Install the package:

```
make install
```

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

6.8. Zlib-1.2.8

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

6.8.1. Installation of Zlib

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.”

6.9. Binutils-2.23.2

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

6.9.1. Installation of Binutils

Apply the following sed for hosts using Texinfo-5.x:

```
sed -i -e 's/@colophon/@@colophon/' \
      -e 's/doc@cygnus.com/doc@@cygnus.com/' bfd/doc/bfd.texinfo
```

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

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

Prepare Binutils for compilation:

```
../binutils-2.23.2/configure --prefix=/tools \
  --build=${CLFS_HOST} --host=${CLFS_TARGET} --target=${CLFS_TARGET} \
  --with-lib-path=/tools/lib --disable-nls --enable-shared \
  --disable-multilib
```

Compile the package:

```
make configure-host
make
```

Install the package:

```
make install
```

Details on this package are located in Section 10.15.2, “Contents of Binutils.”

6.10. GCC-4.8.1

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

6.10.1. Installation of GCC

The following patch contains a number of updates to the 4.8.1 branch by the GCC developers:

```
patch -Np1 -i ../gcc-4.8.1-branch_update-3.patch
```

Make a couple of essential adjustments to the `specs` file to ensure GCC uses our build environment:

```
patch -Np1 -i ../gcc-4.8.1-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"
echo -en '\n#undef STANDARD_STARTFILE_PREFIX_2\n#define STANDARD_STARTFILE_PREFIX_2 "/tools"
```

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-4.8.1/configure --prefix=/tools \
--build=${CLFS_HOST} --host=${CLFS_TARGET} --target=${CLFS_TARGET} \
--libexecdir=/tools/lib --with-local-prefix=/tools --enable-long-long \
--enable-c99 --enable-shared --enable-threads=posix --disable-multilib \
--disable-nls --enable-__cxa_atexit --enable-languages=c,c++ \
--disable-libstdcxx-pch --enable-cloog-backend=isl --with-gmp=/tools \
--with-mpfr=/tools --with-mpc=/tools --with-isl=/tools \
--disable-isl-version-check --with-cloog=/tools --with-system-zlib \
--with-native-system-header-dir=/tools/include --disable-libssp \
--disable-install-libiberty --enable-checking=release \
--enable-libstdcxx-time
```

The meaning of the new configure options:

`--disable-libstdcxx-pch`

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

The following will prevent GCC from looking in the wrong directories for headers and libraries:

```
cp -v Makefile{,.orig}
sed "/^HOST_\(GMP\|ISL\|CLOOG\)\/\(\LIBS\|INC\)\/s:/tools:/cross-tools:g" \
Makefile.orig > Makefile
```

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 10.16.2, “Contents of GCC.”

6.11. Ncurses-5.9

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

6.11.1. Installation of Ncurses

The following patch fixes an issue with some Bash versions:

```
patch -Np1 -i ../ncurses-5.9-bash_fix-1.patch
```

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-debug

This tells Ncurses not to build with debug information.

--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 what type of compiler we are using.

Compile the package:

```
make
```

Install the package:

```
make install
```

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

6.12. Bash-4.2

The Bash package contains the Bourne-Again SHell.

6.12.1. 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-4.2-branch_update-7.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 `eglibc`. 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 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
```

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

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

Details on this package are located in Section 10.35.2, “Contents of Bash.”

6.13. Bison-3.0

The Bison package contains a parser generator.

6.13.1. Installation of Bison

Apply a **sed** which disables the building of `bison.help` when cross-compiling.

```
cp -v Makefile.in{,.orig}
sed '/bison.help:/s/^/# /' Makefile.in.orig > Makefile.in
```

The **configure** script does not determine the correct value for the following. Set the value manually:

```
echo "ac_cv_prog_lex_is_flex=yes" > config.cache
```

Prepare Bison for compilation:

```
M4=m4 ./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 10.27.2, “Contents of Bison.”

6.14. 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**.

6.14.1. Installation of Bzip2

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

```
cp -v Makefile{,.orig}
sed -e 's@^\(all:.*\) test@1@g' 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 10.37.2, “Contents of Bzip2.”

6.15. Coreutils-8.21

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

6.15.1. Installation of Coreutils

Configure can not properly determine how to get free space when cross-compiling - as a result, the **df** program will not be built. Add the following entries to `config.cache` to correct this, and fix various cross-compiling issues:

```
cat > config.cache << EOF
fu_cv_sys_stat_statfs2_bsize=yes
gl_cv_func_working_mkstemp=yes
EOF
```

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 testsuite.

Apply a sed to allow completion of the build:

```
cp -v Makefile{,.orig}
sed -e 's/^#run_help2man\|^run_help2man/#&/' \
  -e 's/^\##run_help2man/run_help2man/' Makefile.orig > Makefile
```

Compile the package:

```
make
```

Install the package:

```
make install
```

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

6.16. Diffutils-3.3

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

6.16.1. 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 10.38.2, “Contents of Diffutils.”

6.17. Findutils-4.4.2

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

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 10.41.2, “Contents of Findutils.”

6.18. File-5.15

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

6.18.1. 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 10.39.2, “Contents of File.”

6.19. Flex-2.5.37

The Flex package contains a utility for generating programs that recognize patterns in text.

6.19.1. Installation of Flex

When cross compiling, the **configure** script does not determine the correct values for the following. Set the values manually:

```
cat > config.cache << EOF
ac_cv_func_malloc_0_nonnull=yes
ac_cv_func_realloc_0_nonnull=yes
EOF
```

Prepare Flex for compilation:

```
M4=m4 ./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 10.29.2, “Contents of Flex.”

6.20. Gawk-4.1.0

The Gawk package contains programs for manipulating text files.

6.20.1. Installation of Gawk

Apply a sed which will allow the build system to complete without error:

```
cp -v extension/Makefile.in{,.orig}
sed -e 's/check-recursive all-recursive: check-for-shared-lib-support/check-recursive
extension/Makefile.in.orig > extension/Makefile.in
```

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 10.40.2, “Contents of Gawk.”

6.21. Gettext-0.18.3.1

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.

6.21.1. Installation of Gettext

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

```
cd gettext-tools
```

When cross-compiling the Gettext configure script assumes we don't have a working `wcwidth` when we do. The following will fix possible compilation errors because of this assumption:

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

Prepare Gettext for compilation:

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

The meaning of the configure options:

--disable-shared

This tells Gettext not to create a shared library.

Compile the package:

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

Install the `msgfmt` binary:

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

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

6.22. Grep-2.14

The Grep package contains programs for searching through files.

6.22.1. Installation of Grep

When cross compiling, the **configure** script does not determine the correct values for the following. Set the values manually:

```
cat > config.cache << EOF
ac_cv_func_malloc_0_nonnull=yes
ac_cv_func_realloc_0_nonnull=yes
EOF
```

Prepare Grep for compilation:

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

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 EGLIBC.

Compile the package:

```
make
```

Install the package:

```
make install
```

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

6.23. Gzip-1.6

The Gzip package contains programs for compressing and decompressing files.

6.23.1. 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 10.46.2, “Contents of Gzip.”

6.24. M4-1.4.17

The M4 package contains a macro processor.

6.24.1. Installation of M4

Configure can not properly determine the results of the following tests:

```
cat > config.cache << EOF
gl_cv_func_btowc_eof=yes
gl_cv_func_mbrtowc_incomplete_state=yes
gl_cv_func_mbrtowc_sanitycheck=yes
gl_cv_func_mbrtowc_null_arg=yes
gl_cv_func_mbrtowc_retval=yes
gl_cv_func_mbrtowc_nul_retval=yes
gl_cv_func_wcrtomb_retval=yes
gl_cv_func_wctob_works=yes
EOF
```

Prepare M4 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 10.26.2, “Contents of M4.”

6.25. Make-3.82

The Make package contains a program for compiling packages.

6.25.1. 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 10.49.2, “Contents of Make.”

6.26. Patch-2.7.1

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

6.26.1. 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 10.53.2, “Contents of Patch.”

6.27. Sed-4.2.2

The Sed package contains a stream editor.

6.27.1. 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 10.17.2, “Contents of Sed.”

6.28. Tar-1.26

The Tar package contains an archiving program.

6.28.1. Installation of Tar

EGLIBC-2.18 does not declare gets():

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

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
ac_cv_func_realloc_0_nonnull=yes
gl_cv_func_mbrtowc_incomplete_state=yes
gl_cv_func_mbrtowc_nul_retval=yes
gl_cv_func_mbrtowc_null_arg=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 10.59.2, “Contents of Tar.”

6.29. Texinfo-4.13a

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

6.29.1. Installation of Texinfo

Prepare Texinfo for compilation:

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

Compile the package:

```
make -C tools/gnulib/lib  
make -C tools  
make
```

Install the package:

```
make install
```

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

6.30. Vim-7.4

The Vim package contains a powerful text editor.

6.30.1. Installation of VIM

The following patch merges all updates from the 7.4 Branch from the Vim developers:

```
patch -Np1 -i ../vim-7.4-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
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-multibyte --enable-gui=no \
  --disable-gtktest --disable-xim --with-features=normal \
  --disable-gpm --without-x --disable-netbeans \
  --with-tlib=ncurses
```

Compile the package:

```
make
```

Install the package:

```
make 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 /etc/vimrc

set nocompatible
set backspace=2
set ruler
syntax on

" End /etc/vimrc
EOF
```

Details on this package are located in Section 10.62.3, “Contents of Vim.”

6.31. XZ Utils-5.0.5

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**.

6.31.1. 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 10.50.2, “Contents of XZ-Utils.”

6.32. 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 chroot into the temporary system.

The boot method is needed when you are building on a different architecture. For example, if you are building a PowerPC system from an x86, you can't chroot. The chroot method is for when you are building on the same architecture. If you are building on, and for, an x86 system, you can simply chroot. The rule of thumb here is if the architectures match and you are running the same series 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.

To chroot, you will also need a Linux Kernel-2.6.32 or greater (having been compiled with GCC-4.1.2 or greater). The reason for the kernel version requirement is that eglibc is built to generate the library for the smallest version of the Linux kernel expected to be supported.

To check your kernel version, run **cat /proc/version** - if it does not say that you are running a 2.6.32 or later Linux kernel, compiled with GCC 4.1.2 or later, you cannot chroot.

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.

There are a few additional packages that will need to be installed to allow you to boot the minimal system. Some of these packages will be installed onto `root` or in `/usr` on the CLFS partition (`${CLFS}/bin`, `${CLFS}/usr/bin`, etc...), rather than `/tools`, using the "DESTDIR" option with `make`. This will require the `clfs` user to have write access to the rest of the CLFS partition, so you will need to temporarily change the ownership of `${CLFS}` to the `clfs` user. Run the following command as `root`:

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

7.2. 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 ${CLFS}/{bin,boot,dev,{etc/,}opt,home,lib,mnt}
mkdir -pv ${CLFS}/{proc,media/{floppy,cdrom},run/{,shm},sbin,srv,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
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}
for dir in ${CLFS}/usr{,/local}; do
    ln -sv share/{man,doc,info} $dir
done
```

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.

7.2.1. FHS Compliance Note

The directory tree is based on the Filesystem Hierarchy Standard (FHS) (available at <http://www.pathname.com/fhs/>). In addition to the 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.

7.3. Creating Essential Symlinks

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

```
ln -sv /tools/bin/{bash,cat,echo,grep,login,passwd,pwd,sleep,stty} ${CLFS}/bin
ln -sv /tools/sbin/{agetty,blkid} ${CLFS}/sbin
ln -sv /tools/bin/file ${CLFS}/usr/bin
ln -sv /tools/lib/libgcc_s.so{,.1} ${CLFS}/usr/lib
ln -sv /tools/lib/libstd*so* ${CLFS}/usr/lib
ln -sv bash ${CLFS}/bin/sh
ln -sv /run ${CLFS}/var/run
```

7.4. Util-linux-2.23.2

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

7.4.1. Installation of Util-linux

Prepare Util-linux for compilation:

```
PKG_CONFIG=true \  
./configure --prefix=/tools --exec-prefix=/tools \  
--build=${CLFS_HOST} --host=${CLFS_TARGET} \  
--disable-makeinstall-chown --disable-login --disable-su
```

Compile the package:

```
make
```

Install the package:

```
make install
```

Details on this package are located in Section 10.20.3, “Contents of Util-linux.”

7.5. Shadow-4.1.5.1

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

7.5.1. Installation of Shadow

Disable the installation of the **groups** program, as Coreutils provides a better version:

```
cp -v src/Makefile.in{,.orig}
sed -e 's/groups$(EXEEXT) //' src/Makefile.in.orig > src/Makefile.in
```

Prepare Shadow for compilation:

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

The meaning of the configure options:

--sysconfdir=/etc

Tells Shadow to install its configuration files into */etc*, rather than */tools/etc*.

Compile the package:

```
make
```

This package does not come with a test suite.

Install the package:

```
make DESTDIR=${CLFS} install
```

Details on this package are located in Section 10.23.4, “Contents of Shadow.”

7.6. E2fsprogs-1.42.8

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

7.6.1. 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:

```
PKG_CONFIG=true \
  ../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.

Compile the package:

```
make LIBUUID="-luuid" STATIC_LIBUUID="-luuid" \
  LIBBLKID="-liblkid" STATIC_LIBBLKID="-liblkid"
```

Install the binaries, documentation and shared libraries:

```
make install
```

Install the static libraries and headers:

```
make install-libs
```

Create needed symlinks for a bootable system:

```
ln -sv /tools/sbin/{fsck.ext2,fsck.ext3,fsck.ext4,e2fsck} ${CLFS}/sbin
```

Details on this package are located in Section 10.22.2, “Contents of E2fsprogs.”

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

The following modifications help locate files specific to this particular build:

```
cp -v src/Makefile{,.orig}
sed -e 's,/usr/lib,/tools/lib,g' \
    src/Makefile.orig > src/Makefile
```

Compile the package:

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

Install the package:

```
make -C src ROOT=${CLFS} install
```

7.7.2. Configuring Sysvinit

Create a new file `${CLFS}/etc/inittab` by running the following:

```
cat > ${CLFS}/etc/inittab << "EOF"
# Begin /etc/inittab

id:3:initdefault:

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

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

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

su:S016:once:/sbin/sulogin

EOF
```

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

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

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

EOF
```

Finally, add the end line to `/${CLFS}/etc/inittab`.

```
cat >> ${CLFS}/etc/inittab << "EOF"
# End /etc/inittab
EOF
```

Details on this package are located in Section 10.58.3, “Contents of Sysvinit.”

7.8. Kmod-15

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

7.8.1. Installation of Kmod

Prepare Kmod for compilation:

```
liblzma_CFLAGS="-I/tools/include" liblzma_LIBS="-L/tools/lib -llzma" \
zlib_CFLAGS="-I/tools/include" zlib_LIBS="-L/tools/lib -lz" \
./configure --prefix=/tools --bindir=/bin \
    --build=${CLFS_HOST} --host=${CLFS_TARGET} --with-xz --with-zlib \
    --disable-manpages
```

Compile the package:

```
make
```

Install the package:

```
make DESTDIR=${CLFS} install
```

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

```
ln -sv kmod ${CLFS}/bin/lsmmod
ln -sv ../bin/kmod ${CLFS}/sbin/depmod
ln -sv ../bin/kmod ${CLFS}/sbin/insmod
ln -sv ../bin/kmod ${CLFS}/sbin/modprobe
ln -sv ../bin/kmod ${CLFS}/sbin/modinfo
ln -sv ../bin/kmod ${CLFS}/sbin/rmmod
```

Details on this package are located in Section 10.52.2, “Contents of Kmod.”

7.9. Eudev-1.3

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

7.9.1. Installation of Eudev

Prepare Eudev for compilation:

```
BLKID_CFLAGS="-I/tools/include" BLKID_LIBS="-L/tools/lib -lblkid" \
KMOD_CFLAGS="-I/tools/include/" KMOD_LIBS="-L/tools/lib -lkmod" \
LDFLAGS="-Wl,-rpath,/tools/lib:/lib" ./configure --prefix=/usr \
  --build=${CLFS_HOST} --host=${CLFS_TARGET} \
  --with-rootprefix='' --enable-split-usr --sysconfdir=/etc \
  --libexecdir=/lib --bindir=/sbin --sbindir=/sbin --libdir=/usr/lib \
  --with-rootlibdir=/lib --disable-introspection --disable-gtk-doc-html \
  --disable-gudev --disable-keymap --with-firmware-path=/lib/firmware \
  --enable-libkmod
```

Compile the package:

```
make
```

Install the package:

```
make DESTDIR=${CLFS} install
```

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

```
install -dv ${CLFS}/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*" \
> ${CLFS}/etc/udev/rules.d/80-net-name-slot.rules
```

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

7.10. Creating the passwd, group, and log 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
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:

```
bin:x:1:1:bin:/bin:/bin/false
```

Can be useful for compatibility with legacy applications.

```
daemon:x:2:6:daemon:/sbin:/bin/false
```

It is often recommended to use an unprivileged User ID/Group ID for daemons to run as, in order to limit their access to the system.

```
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

```
nobody:x:65534:65534:nobody:/:/bin/false
```

Used by NFS

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:
EOF
```

Additional groups you may want to add

```
adm:x:16:root,adm,daemon
```

All users in this group are allowed to do administrative tasks

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

This group has direct access to the console

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

This group is allowed to use the CDRW drive

```
mail:x:30:mail
```

Used by MTAs (Mail Transport Agents)

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

Used by Network News Servers

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

The default GID used by shadow for new users

```
nogroup:x:65533:
```

This is a default group used by some programs that do not require a group

```
nobody:x:65534:
```

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 Eudev 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.linuxbase.org>) recommends only that, besides the group “root” with a Group ID (GID) of 0, a group “bin” with a GID of 1 be present. All other group names and GIDs can be chosen freely by the system administrator since well-written programs do not depend on GID numbers, but rather use the group's name.

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

```
touch ${CLFS}/var/run/utmp ${CLFS}/var/log/{btmp,lastlog,wtmp}
chmod -v 664 ${CLFS}/var/run/utmp ${CLFS}/var/log/lastlog
chmod -v 600 ${CLFS}/var/log/btmp
```

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

7.11. Linux-3.10.14

The Linux package contains the Linux kernel.

7.11.1. 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.

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=powerpc CROSS_COMPILE=${CLFS_TARGET}- menuconfig
```



Warning

Ensure you select all of the necessary mac drivers, particularly for ide and input.

Compile the kernel image and modules:

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

If the use of kernel modules can't be avoided, an `/etc/modprobe.conf` 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. The `modprobe.conf` man page may also be of interest.

Be very careful when reading other documentation relating to kernel modules because it usually applies to 2.4.x kernels only. As far as we know, kernel configuration issues specific to Hotplug and Eudev are not documented. The problem is that Eudev will create a device node only if Hotplug or a user-written script inserts the corresponding module into the kernel, and not all modules are detectable by Hotplug. Note that statements like the one below in the `/etc/modprobe.conf` file do not work with Eudev:

```
alias char-major-XXX some-module
```

Install the modules, if the kernel configuration uses them:

```
make ARCH=powerpc CROSS_COMPILE=${CLFS_TARGET}- \
INSTALL_MOD_PATH=${CLFS} modules_install
```

Install the firmware, if the kernel configuration uses them:

```
make ARCH=powerpc CROSS_COMPILE=${CLFS_TARGET}- \
  INSTALL_FW_PATH=${CLFS}/lib/firmware firmware_install
```

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

Issue the following command to install the kernel:

```
cp -v vmlinux ${CLFS}/boot/clfskernel-3.10.14
```

`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 ${CLFS}/boot/System.map-3.10.14
```

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 ${CLFS}/boot/config-3.10.14
```

Details on this package are located in Section 13.3.2, “Contents of Linux.”

7.12. Hfsutils-3.2.6

The Hfsutils package contains a number of utilities for accessing files on hfs filesystems. It is needed to run **ybin**.

7.12.1. Installation of Hfsutils

If you have created, or will create, the ext2 filesystem on your Mac using ext2fsx you can jump ahead to Section 7.14, “Yaboot-1.3.17.”. The next three packages are for people who cannot do that.

The following patch contains fixes. One that fixes a missing errno.h, the other that allows HFSutils to recognize beyond 2gb devices:

```
patch -Np1 -i ../hfsutils-3.2.6-fixes-1.patch
```

Prepare Hfsutils for compilation:

```
CC="${CC}" ./configure --prefix=/tools
```

Compile the package:

```
make
```

Install the package:

```
make install
```

Details on this package are located in Section 10.63.2, “Contents of Hfsutils.”

7.13. Powerpc-Utills_1.1.3

The Powerpc-Utills package contains a number of utilities for Power Macintoshes and other similar machines. Most of these utilities are now obsolete, but `nvsetenv` is needed by `ybin` to install the bootloader on an hfs partition.

7.13.1. Installation of Powerpc-Utills

This package, originally `pmac-utils`, has issues with NewWorld Macintoshes. The following patch fixes these issues and generally updates the package:

```
patch -Np1 -i ../powerpc-utills_1.1.3-fixes-2.patch
```

This package's Makefile has issues with cross-compiling. Fortunately, we only need one program and it is a simple task to compile it.

```
`${CC}` -o nvsetenv nvsetenv.c nwnvsetenv.c
```

Install the program:

```
install -v -m755 nvsetenv /tools/sbin
```

Details on this package are located in Section 10.65.2, “Contents of Powerpc-Utills.”

7.14. Yaboot-1.3.17

The Yaboot package contains a PowerPC Boot Loader for machines using Open Firmware such as NewWorld Macintoshes.

7.14.1. Installation of Yaboot

The following patch adds stub functions for newer e2fsprogs releases.

```
patch -Np1 -i ../yaboot-1.3.17-stubfuncs-1.patch
```

The following patch allows **ofpath** to use `PATH_PREFIX` like the other `ybin` scripts:

```
patch -Np1 -i ../yaboot-1.3.17-ofpath_path_prefix-1.patch
```

The Makefile is already set to do kernel-style cross-compiling, but it will try to use **strip** to strip the second-stage loader. It also tries to change user and group ownership for the installed files to `root`, which the `clfs` user cannot do. It will also fail due to a compile warning with the `-Werror` flag on. The following command fixes these issues:

```
cp -v Makefile{,.orig}
sed -e "s/\\(strip \\)/${CLFS_TARGET}-\\1/" \\
    -e 's/-o root -g root//' \\
    -e 's/-Werror//' \\
    Makefile.orig > Makefile
```

Compile the package:

```
make CROSS=${CLFS_TARGET}-
```

Install the package:

```
make CROSS=${CLFS_TARGET}- ROOT=/tools PREFIX="" install
```

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

7.15. 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 `.bash_profile` file. Create the `.bash_profile` file 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.

7.16. 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
proc          /proc         proc   defaults                0     0
sysfs         /sys          sysfs  defaults                0     0
devpts        /dev/pts      devpts gid=5,mode=620         0     0
shm           /dev/shm      tmpfs  defaults                0     0
tmpfs         /run          tmpfs  defaults                0     0
devtmpfs      /dev          devtmpfs mode=0755,nosuid      0     0

# End /etc/fstab
EOF
```

Replace `[xxx]`, `[yyy]`, and `[fff]` with the values appropriate for the system, for example, `hda2`, `hda5`, and `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. Bootscripts for CLFS 2.1-pre1

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

7.17.1. Installation of Bootscripts

Install the package:

```
make DESTDIR=${CLFS} install-minimal
```

The **setclock** script reads the time from the hardware clock, also known as the BIOS or the Complementary Metal Oxide Semiconductor (CMOS) clock. If the hardware clock is set to UTC, this script will convert the hardware clock's time to the local time using the `/etc/localtime` file (which tells the **hwclock** program which timezone the user is in). There is no way to detect whether or not the hardware clock is set to UTC, so this needs to be configured manually.

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 `/etc/sysconfig/clock` script. 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 > ${CLFS}/etc/sysconfig/clock << "EOF"
# Begin /etc/sysconfig/clock

UTC=1

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

Details on this package are located in Section 11.2.2, “Contents of Bootscripts.”

7.18. Populating /dev

7.18.1. Creating Initial Device Nodes



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.

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

Before `udev` starts, a `tmpfs` filesystem is mounted over `/dev` and the previous entries are no longer available. The following command creates files that are copied over by the `udev` bootscript:

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

7.19. Changing Ownership

Currently, the `${CLFS}` directory and all of its subdirectories are owned by the user `clfs`, a user that exists only on the host system. For security reasons, the `${CLFS}` root directory and all of its subdirectories should be owned by `root`. Change the ownership for `${CLFS}` and its subdirectories by running this command:

```
chown -Rv 0:0 ${CLFS}
```

The following files are to be owned by the group `utmp` not by `root`.

```
chgrp -v 13 ${CLFS}/var/run/utmp ${CLFS}/var/log/lastlog
```

7.20. Making the Temporary System Bootable

Some of the idiosyncracies of booting on `ppc` are discussed in Appendix E. Essentially, there are two options here - either copy the bootloader to an `OSX` root partition and boot from Open Firmware, or use an `install`, `Live`, or `rescue` CD to set up a bootstrap partition.

7.20.1. Copying the bootloader to OSX and booting from OF.

You must now ensure that `/tools/etc/yaboot.conf` contains the correct details for the `CLFS` system. Consult Section 13.4, "Making the `CLFS` System Bootable." for details, but note that at this point you do not need the `install`, `magicboot`, `enablecdboot` or `macosx` parameters because these are not available when you boot from Open Firmware.

By this stage, you should have the temporary system on an `ext2` filesystem on your Mac. Now, from within `OSX`, copy `/tools/lib/yaboot/yaboot`) and `/tools/etc/yaboot.conf`) to the `OSX /` directory.

Each time you want to boot into the temporary system, hold down the option-command-o-f keys to get to Open Firmware, then use the following command, replacing X with the number of the partition containing the *OSX* root filesystem (typically, this will be '3').

```
boot hd:X,yaboot
```

7.20.2. Using a CD to set up the bootstrap partition.

This is particularly appropriate if you cannot write to an ext2 filesystem from OSX. Boot from the CD, and (as necessary) create partitions and filesystems, mount the CLFS partition at `/tools` and untar the temporary system there.

Now set up `/tools/etc/yaboot.conf` - see Section 13.4, “Making the CLFS System Bootable.” for details of what should be in it, but note that the *install* and *magicboot* specifications should point to `/tools/lib/yaboot/` and *not* `/usr/lib/yaboot`.

To write the bootloader to the disk, with `/tools/sbin` first on your path and `/proc` mounted, run the following command:



Warning

The following command will update the bootstrap partition and the boot variable in Open Firmware. Do not run the command if this is not desired.

```
PATH_PREFIX=/tools ybin -v -C /tools/etc/yaboot.conf
```

Alternatively, if the bootstrap partition has not already been initialized, perhaps because you are using a Live CD, you will need to use a different command to install the bootloader for the first time:

```
PATH_PREFIX=/tools mkofboot
```

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. Util-linux-2.23.2

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

8.2.1. Installation of Util-linux

Prepare Util-linux for compilation:

```
./configure --prefix=/tools \  
  --build=${CLFS_HOST} --host=${CLFS_TARGET} \  
  --disable-makeinstall-chown --disable-su --disable-login
```

Compile the package:

```
make
```

Install the package:

```
make install
```

Details on this package are located in Section 10.20.3, “Contents of Util-linux.”

8.3. 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,sys}
```

Now mount the file systems:

```
mount -vt proc proc ${CLFS}/proc
mount -vt sysfs sysfs ${CLFS}/sys
```

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.

Two device nodes, `/dev/console` and `/dev/null`, are required to be present on the filesystem. These are needed by the kernel even before starting Eudev 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 are created by the Eudev package. Since this package is not available to us right now, we must take other steps to provide device nodes under on the CLFS filesystem. We will use the “bind” option in the mount command to make our host system's `/dev` structure appear in the new CLFS filesystem:

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

Additional file systems will soon be mounted from within the chroot environment. To keep the host up to date, perform a “fake mount” for each of these now:

```
if [ -h ${CLFS}/dev/shm ]; then
    link=$(readlink ${CLFS}/dev/shm)
    mkdir -p ${CLFS}/$link
    mount -f -vt tmpfs shm ${CLFS}/$link
    unset link
else
    mount -f -vt tmpfs shm ${CLFS}/dev/shm
fi
mount -f -vt devpts -o gid=5,mode=620 devpts ${CLFS}/dev/pts
```

8.4. Before Entering the Chroot Environment

8.4.1. 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 10.34, “Automake-1.12.4”. 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 10.34, “Automake-1.12.4” tarball and **cd** into the created directory. Then execute the following to see what the detected target triplet is by **config.guess**:

```
build-aux/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 8.5, “Entering the Chroot Environment”.

8.4.2. Using Setarch

If your host has a tool called **setarch** this may solve your problems. The reason for saying may is because on a 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 10.34, “Automake-1.12.4” directory:

```
setarch linux32 build-aux/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 8.5, “Entering the Chroot Environment”. If not, there is one more option covered in this book.

8.4.3. 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://cross-lfs.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=ppc
```

The meaning of the **make** and **install** options:

```
uname_hack_fake_machine=ppc
```

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 10.34, “Automake-1.12.4” directory:

```
build-aux/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 1.6, “Help” for more information.

8.5. 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.

8.6. Changing Ownership



Note

This step is not optional as some of the binaries in `/tools` are set `u+s`. Leaving the permissions as is could cause some commands, `mount` in particular, to fail later.

Currently, the `/tools` and `/cross-tools` directories are owned by the user `clfs`, a user that exists only on the host system. Although the `/tools` and `/cross-tools` directories 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 the `/tools` directory and all the files therein, thus exposing these files to possible malicious manipulation.

To avoid this issue, 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.

8.7. 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,srv,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
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}
for dir in /usr{,/local}; do
  ln -sv share/{man,doc,info} $dir
done
```

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.

8.7.1. FHS Compliance Note

The directory tree is based on the Filesystem Hierarchy Standard (FHS) (available at <http://www.pathname.com/fhs/>). In addition to the 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.

8.8. Creating Essential Symlinks

Some programs use hard-wired paths to programs which do not exist yet. In order to satisfy these programs, create a number of symbolic links which will be replaced by real files throughout the course of 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/libstd* /usr/lib
ln -sv bash /bin/sh
ln -sv /run /var/run
```

8.9. Creating the passwd, group, and log 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
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:

```
bin:x:1:1:bin:/bin:/bin/false
```

Can be useful for compatibility with legacy applications.

```
daemon:x:2:6:daemon:/sbin:/bin/false
```

It is often recommended to use an unprivileged User ID/Group ID for daemons to run as, in order to limit their access to the system.

```
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

```
nobody:x:65534:65534:nobody:/:/bin/false
```

Used by NFS

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:
EOF
```

Additional groups you may want to add

```
adm:x:16:root,adm,daemon
```

All users in this group are allowed to do administrative tasks

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

This group has direct access to the console

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

This group is allowed to use the CDRW drive

```
mail:x:30:mail
```

Used by MTAs (Mail Transport Agents)

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

Used by Network News Servers

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

The default GID used by shadow for new users

```
nogroup:x:65533:
```

This is a default group used by some programs that do not require a group

```
nobody:x:65534:
```

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 Eudev 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.linuxbase.org>) recommends only that, besides the group “root” with a Group ID (GID) of 0, a group “bin” with a GID of 1 be present. All other group names and GIDs can be chosen freely by the system administrator since well-written programs do not depend on GID numbers, but rather use the group's name.

To remove the “I have no name!” prompt, start a new shell. Since a full Glibc was installed in 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.

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

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

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

8.10. Mounting Kernel Filesystems

8.10.1. Mounting Additional Kernel Filesystems

Mount the proper virtual (kernel) file systems on the newly-created directories:

```
mount -vt devpts -o gid=5,mode=620 none /dev/pts
mount -vt tmpfs none /dev/shm
```

The **mount** commands executed above may result in the following warning message:

```
can't open /etc/fstab: No such file or directory.
```

This file—`/etc/fstab`—has not been created yet (unless using the boot method), but is also not required for the file systems to be properly mounted. The warning can be safely ignored.

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 and Binutils test suites. Check is needed for KBD tests. Installing four packages for testing purposes may seem excessive, but it is very reassuring, if not essential, to know that the most important tools are working properly.

9.2. Tcl-8.6.1

The Tcl package contains the Tool Command Language.

9.2.1. Installation of Tcl

Increase memory size for regular expressions where required.

```
sed -i s/500/5000/ generic/regc_nfa.c
```

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-1.5.1

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

9.5. Check-0.9.10

The Check package is a unit testing framework for C.

9.5.1. Installation of Check

Prepare Check for compilation:

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

Build the package:

```
make
```

Install the package:

```
make install
```

9.5.2. 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

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.cross-lfs.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.cross-lfs.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 testsuites. 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.cross-lfs.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://cross-lfs.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-5.18.1

The Perl package contains the Practical Extraction and Report Language.

10.4.1. Installation of Perl

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

```
patch -Np1 -i ../perl-5.18.1-libc-1.patch
```

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 10.31.2, “Contents of Perl.”

10.5. Linux-Headers-3.10.14

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

10.5.1. Installation of Linux-Headers

For this step you will need the kernel tarball.

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

Short Descriptions

/usr/include/{asm,asm-generic,drm,linux,mtd,rdma,sound,video}/*.h	The Linux API headers
---	-----------------------

10.6. Man-pages-3.54

The Man-pages package contains over 1,200 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. EGLIBC-2.18

The EGLIBC 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 EGLIBC



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.” EGLIBC 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 will attempt to test for a library that is only used in the test suite and is never installed. Prevent the script from testing for this library with the following command:

```
sed -i 's/\(&& $name ne\) "db1"/ & \1 "nss_test1"/' scripts/test-installation.pl
```

This same script performs its tests by attempting to compile test programs against certain libraries. However it does not specify the `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 EGLIBC 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 EGLIBC install because the EGLIBC Autoconf tests would give false results and defeat the goal of achieving a clean build.

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

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

Prepare EGLIBC for compilation:

```
../eglibc-2.18/configure --prefix=/usr \
  --disable-profile --enable-kernel=2.6.32 --libexecdir=/usr/lib/eglibc \
  --enable-obsolete-rpc
```

The meaning of the new configure option:

```
--libexecdir=/usr/lib/eglibc
```

This changes the location of the `getconf` utility from its default of `/usr/libexec` to `/usr/lib/eglibc`.

Compile the package:

```
make
```



Important

The test suite for EGLIBC is considered critical. Do not skip it under any circumstance.

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

```
cp -v ../eglibc-2.18/iconvdata/gconv-modules iconvdata
make -k check 2>&1 | tee eglibc-check-log; grep Error eglibc-check-log
```

The EGLIBC test suite is highly dependent on certain functions of the host system, in particular the kernel. The `posix/annexc` test normally fails and you should see `Error 1 (ignored)` in the output. Apart from this, the EGLIBC 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 *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 2.4, “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.

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

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

Install the package:

```
make install
```

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 `eglibc-2.18/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 EGLIBC 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 package) does not work properly in them. Various attempts to circumvent these restrictions are documented in internationalization-related hints.

10.7.3. Configuring EGLIBC

The */etc/nsswitch.conf* file needs to be created because, although EGLIBC provides defaults when this file is missing or corrupt, the EGLIBC 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 ../tzdata2013g.tar.gz

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

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

cp -v zone.tab iso3166.tab $ZONEINFO
zic -d $ZONEINFO -p America/New_York
unset ZONEINFO
```

The meaning of the `zic` commands:

```
zic -L /dev/null ...
```

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

```
zic -L leapseconds ...
```

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

```
zic ... -p ...
```

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

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 --remove-destination /usr/share/zoneinfo/[xxx] \
  /etc/localtime
```

Replace `[xxx]` with the name of the time zone that **tzselect** provided (e.g., *Canada/Eastern*).

The meaning of the `cp` option:

```
--remove-destination
```

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

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 EGLIBC

Installed programs:	catchsegv, gencat, getconf, getent, iconv, iconvconfig, ldconfig, ldd, lddlibc4, locale, localedef, makedb, mtrace, nscd, pccprofiledump, pldd, rpcgen, sln, sprof, tzselect, xtrace, zdump, and zic
Installed libraries:	ld.so, libBrokenLocale.[a,so], libSegFault.so, libanl.[a,so], libbsd-compat.a, libc.[a,so], libc_nonshared.a, libcidn.[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, and libutil.[a,so]
Installed directories:	/usr/include/arpa, /usr/include/bits, /usr/include/gnu, /usr/include/net, /usr/include/netash, /usr/include/netatalk, /usr/include/netax25, /usr/include/neteconet, /usr/include/netinet, /usr/include/netipx, /usr/include/netiucv, /usr/include/netpacket, /usr/include/netrom, /usr/include/netrose, /usr/include/nfs, /usr/include/protocols, /usr/include/rpc, /usr/include/rpcsvc, /usr/include/sys, /usr/lib/gconv, /usr/lib/eglibc, /usr/lib/locale, /usr/share/i18n, /usr/share/zoneinfo, /var/cache/ldconfig

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
nscd	A daemon that provides a cache for the most common name service requests
pccprofiledump	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
libbsd-compat	Provides the portability needed in order to run certain Berkeley Software Distribution (BSD) programs under Linux
libc	The main C library
libcidn	Used internally by EGLIBC 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 <code>g++</code>
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 EGLIBC, 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 amend the GCC specs file so that it points 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
```

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

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



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 '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. GMP-5.1.3

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

10.9.1. 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
```

Compile the package:

```
make
```



Important

The test suite for GMP is considered critical. Do not skip it under any circumstance.

Test the results:

```
make check
```

Install the package:

```
make install
```

10.9.2. Contents of GMP

Installed libraries: libgmp.[a,so], libgmpxx.[a,so], libmp.[a,so]

Short Descriptions

`libgmp` Contains the definitions for GNU multiple precision functions.
`libgmpxx` Contains a C++ class wrapper for GMP types.
`libmp` Contains the Berkeley MP compatibility library.

10.10. MPFR-3.1.2

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

10.10.1. Installation of MPFR

Prepare MPFR for compilation:

```
CC="gcc -isystem /usr/include" \
LDLDFLAGS="-Wl,-rpath-link,/usr/lib:/lib" \
./configure --prefix=/usr --enable-shared \
--with-gmp=/usr
```

Compile the package:

```
make
```



Important

The test suite for MPFR is considered critical. Do not skip it under any circumstance.

Test the results:

```
make check
```

Install the package:

```
make install
```

10.10.2. Contents of MPFR

Installed libraries:	libmpfr.[a,so]
Installed directory:	/usr/share/doc/mpfr

Short Descriptions

`libmpfr` The Multiple Precision Floating-Point Reliable Library.

10.11. MPC-1.0.1

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

10.11.1. Installation of MPC

Prepare MPC for compilation:

```
CC="gcc -isystem /usr/include" \
LDFLAGS="-Wl,-rpath-link,/usr/lib:/lib" \
./configure --prefix=/usr
```

Compile the package:

```
make
```



Important

The test suite for MPC is considered critical. Do not skip it under any circumstance.

Test the results:

```
make check
```

Install the package:

```
make install
```

10.11.2. Contents of MPC

Installed libraries: libmpc.[a,so]

Short Descriptions

`libmpc` The Multiple Precision Complex Library.

10.12. ISL-0.12.1

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

10.12.1. Installation of ISL

Prepare ISL for compilation:

```
CC="gcc -isystem /usr/include" \
LD_FLAGS="-Wl,-rpath-link,/usr/lib:/lib" \
./configure --prefix=/usr
```

Compile the package:

```
make
```



Important

The test suite for ISL is considered critical. Do not skip it under any circumstance.

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/*gdb.py /usr/share/gdb/auto-load/usr/lib
```

10.12.2. Contents of ISL

Installed libraries: libisl.[a,so]

Short Descriptions

`libisl` The Integer Set Library.

10.13. CLoog-0.18.0

CLoog is a library to generate code for scanning Z-polyhedra. In other words, it finds code that reaches each integral point of one or more parameterized polyhedra. GCC links with this library in order to enable the new loop generation code known as Graphite.

10.13.1. Installation of CLoog

Prepare CLoog for compilation:

```
CC="gcc -isystem /usr/include" \
LDFLAGS="-Wl,-rpath-link,/usr/lib:/lib" \
./configure --prefix=/usr --enable-shared --with-isl=system
```

Compile the package:

```
make
```



Important

The test suite for CLoog is considered critical. Do not skip it under any circumstance.

Test the results:

```
make check
```

Install the package:

```
make install
```

10.13.2. Contents of CLoog

Installed program:	cloog
Installed libraries:	libcloog-isl.[a,so]
Installed directories:	/usr/include/cloog

Short Descriptions

cloog	Loop generator for scanning Z-polyhedra
libcloog-isl	Isl backend for CLoog.

10.14. Zlib-1.2.8

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 it into `/lib` and then relink it to `/usr/lib`:

```
mv -v /usr/lib/libz.so.* /lib
ln -svf ../../lib/libz.so.1 /usr/lib/libz.so
```

10.14.2. Contents of Zlib

Installed libraries: `libz.[a,so]`

Short Descriptions

`libz` Contains compression and decompression functions used by some programs

10.15. Binutils-2.23.2

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

10.15.1. 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-2.23.2/configure --prefix=/usr \
  --enable-shared
```

Compile the package:

```
make configure-host
```



Important

During **make configure-host** you may receive the following error message. It is safe to ignore.

```
WARNING: `flex' is missing on your system. You should only
need it if you modified a `.l' file. You may need the `Flex'
package in order for those modifications to take effect. You
can get `Flex' from any GNU archive site.
```

```
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**

The test suite for Binutils is considered critical. Do not skip it under any circumstance.

Test the results:

```
make check
```

Install the package:

```
make tooldir=/usr install
```

Install the `libiberty` header file that is needed by some packages:

```
cp -v ../binutils-2.23.2/include/libiberty.h /usr/include
```

10.15.2. Contents of Binutils

Installed programs:	<code>addr2line</code> , <code>ar</code> , <code>as</code> , <code>c++filt</code> , <code>elfedit</code> , <code>gprof</code> , <code>ld</code> , <code>ld.bfd</code> , <code>nm</code> , <code>objcopy</code> , <code>objdump</code> , <code>ranlib</code> , <code>readelf</code> , <code>size</code> , <code>strings</code> , and <code>strip</code>
Installed libraries:	<code>libiberty.a</code> , <code>libbfd.[a,so]</code> , and <code>libopcodes.[a,so]</code>
Installed directory:	<code>/usr/lib/ldscripts</code>

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 <code>gcc</code> 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 <code>ld</code>
nm	Lists the symbols occurring in a given object file
objcopy	Translates one type of object file into another
objdump	Displays information about the given object file, with options controlling the particular information to display; the information shown is useful to programmers who are working on the compilation tools
ranlib	Generates an index of the contents of an archive and stores it in the archive; the index lists all of the symbols defined by archive members that are relocatable object files
readelf	Displays information about ELF type binaries
size	Lists the section sizes and the total size for the given object files

strings	Outputs, for each given file, the sequences of printable characters that are of at least the specified length (defaulting to four); for object files, it prints, by default, only the strings from the initializing and loading sections while for other types of files, it scans the entire file
strip	Discards symbols from object files
libiberty	Contains routines used by various GNU programs, including getopt , obstack , strerror , strtol , and strtoul
libbfd	The Binary File Descriptor library
libopcodes	A library for dealing with opcodes—the “readable text” versions of instructions for the processor; it is used for building utilities like objdump .

10.16. GCC-4.8.1

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

10.16.1. Installation of GCC

The following patch contains a number of updates to the 4.8.1 branch by the GCC developers:

```
patch -Np1 -i ../gcc-4.8.1-branch_update-3.patch
```

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
```

Apply a **sed** substitution that will suppress the installation of **libiberty.a**. The version of **libiberty.a** provided by Binutils will be used instead:

```
sed -i 's/install_to_$(INSTALL_DEST) //' libiberty/Makefile.in
```

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:

```
CC="gcc -isystem /usr/include" \
CXX="g++ -isystem /usr/include" \
LDFLAGS="-Wl,-rpath-link,/usr/lib:/lib" \
  ../gcc-4.8.1/configure --prefix=/usr \
  --libexecdir=/usr/lib --enable-shared --enable-threads=posix \
  --enable-__cxa_atexit --enable-c99 --enable-long-long \
  --enable-clocale=gnu --enable-languages=c,c++ \
  --disable-multilib --disable-libstdcxx-pch \
  --enable-cloog-backend=isl --disable-isl-version-check --with-system-zlib \
  --enable-checking=release --enable-libstdcxx-time \
  --disable-install-libiberty
```

Compile the package:

```
make
```



Important

The test suite for GCC is considered critical. Do not skip it under any circumstance.

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-4.8.1/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
```

Some packages expect the C preprocessor to be installed in the `/lib` directory. To support those packages, create this symlink:

```
ln -sv ../usr/bin/cpp /lib
```

Many packages use the name `cc` to call the C compiler. To satisfy those packages, create a symlink:

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

Finally, move a misplaced file:

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

10.16.2. Contents of GCC

Installed programs:	<code>c++</code> , <code>cc</code> (link to <code>gcc</code>), <code>cpp</code> , <code>g++</code> , <code>gcc</code> , and <code>gcov</code>
Installed libraries:	<code>libgcc.a</code> , <code>libgcc_eh.a</code> , <code>libgcc_s.so</code> , <code>libgcov.a</code> , <code>libgomp.[a,so]</code> , <code>libmudflap.[a,so]</code> , <code>libmudflapth.[a,so]</code> , <code>libssp.[a,so]</code> , <code>libssp_nonshared.a</code> , <code>libstdc++.a</code> , <code>libstdc++.a</code> , and <code>libsupc++</code>
Installed directories:	<code>/usr/include/c++</code> , <code>/usr/lib/gcc</code> , <code>/usr/share/gcc-4.8.1</code>

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
libgcc	Contains run-time support for <code>gcc</code>
libgcov	Library that is linked into a program when <code>gcc</code> is instructed to enable profiling
libgomp	GNU implementation of the OpenMP API for multi-platform shared-memory parallel programming in C/C++ and Fortran

<code>libmudflap</code>	The <code>libmudflap</code> libraries are used by GCC for instrumenting pointer and array dereferencing operations.
<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

10.17. Sed-4.2.2

The Sed package contains a stream editor.

10.17.1. Installation of Sed

Prepare Sed for compilation:

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

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 -C doc install-html
```

10.17.2. Contents of Sed

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

Short Descriptions

sed Filters and transforms text files in a single pass

10.18. Ncurses-5.9

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

10.18.1. Installation of Ncurses

The following patch contains updates from the 5.9 branch by the Ncurses developers:

```
patch -Np1 -i ../ncurses-5.9-branch_update-4.patch
```

Prepare Ncurses for compilation:

```
./configure --prefix=/usr --libdir=/lib \
  --with-shared --without-debug --enable-widex \
  --with-manpage-format=normal \
  --with-default-terminfo-dir=/usr/share/terminfo
```

Compile the package:

```
make
```

This package has a test suite, and can be ran 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 Ncurses static libraries to the proper location:

```
mv -v /lib/lib{panelw,menuw,formw,ncursesw,ncurses++w}.a /usr/lib
```

Create symlinks in `/usr/lib`:

```
rm -v /lib/lib{ncursesw,menuw,panelw,formw}.so
ln -svf ../../lib/libncursesw.so.5 /usr/lib/libncursesw.so
ln -svf ../../lib/libmenuw.so.5 /usr/lib/libmenuw.so
ln -svf ../../lib/libpanelw.so.5 /usr/lib/libpanelw.so
ln -svf ../../lib/libformw.so.5 /usr/lib/libformw.so
```

Now we will make our Ncurses compatible for older and non-widex compatible programs can build properly:

```
for lib in curses 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 libncursesw.so /usr/lib/libcursesw.so
ln -sfv libncursesw.a /usr/lib/libcursesw.a
ln -sfv libncurses++w.a /usr/lib/libncurses++.a
ln -sfv ncursesw5-config /usr/bin/ncurses5-config
```

Now we will create a symlink for `/usr/share/terminfo` in `/usr/lib` for compatibility:

```
ln -sfv ../share/terminfo /usr/lib/terminfo
```

10.18.2. Contents of Ncurses

Installed programs:	captoinfo (link to tic), clear, infocmp, infotocap (link to tic), ncursesw5-config, reset (link to tset), tabs, tic, toe, tput, and tset
Installed libraries:	libcursesw.so (link to libncursesw.so), libformw.[a,so], libmenuw.[a,so], libncursesw.a, libncursesw.[a,so], and 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
ncursesw5-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

10.19. 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.

10.19.1. Installation of Pkg-config-lite

Prepare Pkg-config-lite for compilation:

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

Compile the package:

```
make
```

To test the results, issue: **make check**.

Install the package:

```
make install
```

10.19.2. Contents of Pkg-config-lite

Installed programs:	pkg-config
Installed directory:	/usr/share/doc/pkg-config

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.

10.20. Util-linux-2.23.2

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

10.20.1. FHS compliance notes

The FHS recommends using the `/var/lib/hwclock` directory instead of the usual `/etc` directory as the location for the `adjtime` file. To make the `hwclock` program FHS-compliant, run the following:

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

10.20.2. Installation of Util-linux

Prepare Util-linux for compilation:

```
./configure --enable-write --disable-login --disable-su
```

The meaning of the configure options:

`--enable-write`

This option allows the `write` program to be installed.

`--disable-login --disable-su`

Disables building the `login` and `su` programs, as the Shadow package installs its own versions.

Compile the package:

```
make
```

To test the results, issue: `make check`.

Install the package:

```
make install
```

Move the `logger` binary to `/bin` as it is needed by the CLFS-Bootscripts package:

```
mv -v /usr/bin/logger /bin
```

10.20.3. Contents of Util-linux

Installed programs:

addpart, agetty, blkid, blockdev, cal, cfdisk, chcpu, chrt, col, colcrt, colrm, column, ctrlaltdel, cytune, delpart, dmesg, eject, fallocate, fdformat, fdisk, findfs, findmnt, flock, fsck, fsck.cramfs, fsck.minix, fsfreeze, fstrim, getopt, hexdump, hwclock, ionice, ipcmk, ipcrm, ipcs, isosize, kill, ldattach, logger, look, losetup, lsblk, lscpu, lslocks, mcookie, mkfs, mkfs.bfs, mkfs.cramfs, mkfs.minix, mkswap, more, mount, mountpoint, namei, partx, pg, pivot_root, prlimit, raw, readprofile, rename, renice, resizepart, rev, rtcwake, script, scriptreplay, setarch, setsid, setterm, sfdisk, sulogin, swaptlabel, swapoff (link to swapon), swapon, switch_root, tailf, taskset, tunelp, ul, umount, unshare, utmpdump, uidd, uuidgen, wall, wdctl, whereis, wpefs, and write libblkid.[a,so], libmount.[a,so], and libuuid.[a,so]

Installed libraries:

Installed directories:

/usr/include/blkid, /usr/include/libmount, /usr/include/uuid, /usr/share/getopt, /var/lib/hwclock

Short Descriptions

addpart	Informs the kernel of a new partition
agetty	Opens a tty port, prompts for a login name, and then invokes the login program
blkid	A command line utility to locate and print block device attributes
blockdev	Allows users to call block device ioctls from the command line
cal	Displays a simple calendar
cfdisk	Manipulates the partition table of the given device
chcpu	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
cytune	Tunes the parameters of the serial line drivers for Cyclades cards
ddate	Gives the Discordian date or converts the given Gregorian date to a Discordian one
delpart	Asks the 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
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
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
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	Prints or changes the label or UUID of a swap area
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
tunelp	Tunes the parameters of the line printer
ul	A filter for translating underscores into escape sequences indicating underlining for the terminal in use
umount	Disconnects a file system from the system's file tree
unshare	Runs a program with some namespaces unshared from parent
utmpdump	Displays the content of the given login file in a more user-friendly format
uuid	A daemon used by the UUID library to generate time-based UUIDs in a secure and guranteed-unique fashion.
uuidgen	Creates new UUIDs. Each new UUID can reasonably be considered unique among all UUIDs created, on the local system and on other systems, in the past and in the future
wall	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

10.21. Procps-3.2.8

The Procps package contains programs for monitoring processes.

10.21.1. Installation of Procps

The following patch adds process control group support to ps:

```
patch -Np1 -i ../procps-3.2.8-ps_cgroup-1.patch
```

The following patch fixes an issue where some procps utils print an error on the screen if the monitor isn't running at 60Hz:

```
patch -Np1 -i ../procps-3.2.8-fix_HZ_errors-1.patch
```

The following fixes an issue with Make 3.82:

```
sed -i -r '/^-include/s/\*(.*)/proc\1 ps\1/' Makefile
```

Compile the package:

```
make
```

This package does not come with a test suite.

Install the package:

```
make SKIP='/bin/kill /usr/share/man/man1/kill.1' install
```

10.21.2. Contents of Procps

Installed programs:	free, pgrep, pkill, pmap, ps, pwdx, skill, slabtop, snice, sysctl, tload, top, uptime, vmstat, w, and watch
Installed library:	libproc.so

Short Descriptions

free	Reports the amount of free and used memory (both physical and swap memory) in the system
pgrep	Looks up processes based on their name and other attributes
pkill	Signals processes based on their name and other attributes
pmap	Reports the memory map of the given process
ps	Lists the current running processes
pwdx	Reports the current working directory of a process
skill	Sends signals to processes matching the given criteria
slabtop	Displays detailed kernel slab cache information in real time
snice	Changes the scheduling priority of processes matching the given criteria
sysctl	Modifies kernel parameters at run time
tload	Prints a graph of the current system load average

top	Displays a list of the most CPU intensive processes; it provides an ongoing look at processor activity in real time
uptime	Reports how long the system has been running, how many users are logged on, and the system load averages
vmstat	Reports virtual memory statistics, giving information about processes, memory, paging, block Input/Output (IO), traps, and CPU activity
w	Shows which users are currently logged on, where, and since when
watch	Runs a given command repeatedly, displaying the first screen-full of its output; this allows a user to watch the output change over time
libproc	Contains the functions used by most programs in this package

10.22. E2fsprogs-1.42.8

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

10.22.1. Installation of E2fsprogs

The E2fsprogs documentation recommends that the package be built in a subdirectory of the source tree:

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

Prepare E2fsprogs for compilation:

```
../configure --prefix=/usr --with-root-prefix="" \
  --enable-elf-shlibs --disable-libblkid \
  --disable-libuuid --disable-fsck \
  --disable-uuid
```

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.

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
```

10.22.2. Contents of E2fsprogs

Installed programs:	badblocks, chattr, compile_et, debugfs, dumpe2fs, e2freefrag, e2fsck, e2image, e2initrd_helper, e2label, e2undo, e4defrag, filefrag, fsck.ext2, fsck.ext3, fsck.ext4, fsck.ext4dev, logsave, lsattr, mk_cmds, mke2fs, mkfs.ext2, mkfs.ext3, mkfs.ext4, mkfs.ext4dev, mklost+found, resize2fs, and tune2fs
Installed libraries:	libcom_err.[a,so], libe2p.[a,so], libext2fs.[a,so], libss.[a,so], and 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

libquota

Provides an interface for creating and updating quota files and ext4 superbloc fields

libss

Used by **debugfs**

10.23. Shadow-4.1.5.1

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

10.23.1. Installation of Shadow



Note

If you would like to enforce the use of strong passwords, refer to <http://cblfs.cross-lfs.org/index.php/Cracklib> for installing Cracklib prior to building Shadow. Then add `--with-libcrack` to the **configure** command below.

Disable the installation of the **groups** program and its man pages, as Coreutils provides a better version:

```
sed -i 's/groups$(EXEEXT) //' src/Makefile.in
find man -name Makefile.in -exec sed -i '/groups\.1\.xml/d' '{}' \;
find man -name Makefile.in -exec sed -i 's/groups\.1 / /' {} \;
```

Prepare Shadow for compilation:

```
./configure --sysconfdir=/etc
```

The meaning of the configure options:

```
--sysconfdir=/etc
```

Tells Shadow to install its configuration files into `/etc`, rather than `/usr/etc`.

Compile the package:

```
make
```

This package does not come with a test suite.

Install the package:

```
make install
```

Instead of using the default `crypt` 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@'
```



Note

If you built Shadow with Cracklib support, execute this `sed` to correct the path to the Cracklib dictionary:

```
sed -i 's@DICTPATH.*@DICTPATH\t/lib/cracklib/pw_dict@' /etc/login.defs
```

Move a misplaced program to its proper location:

```
mv -v /usr/bin/passwd /bin
```

10.23.2. 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.cross-lfs.org/index.php/Configuring_for_Adding_Users for more information.

10.23.3. Setting the root password

Choose a password for user `root` and set it by running:

```
passwd root
```

10.23.4. Contents of Shadow

Installed programs:	<code>chage</code> , <code>chfn</code> , <code>chpasswd</code> , <code>chgrp</code> , <code>chsh</code> , <code>expiry</code> , <code>faillog</code> , <code>gpasswd</code> , <code>groupadd</code> , <code>groupdel</code> , <code>groupmems</code> , <code>groupmod</code> , <code>grpck</code> , <code>grpconv</code> , <code>grpunconv</code> , <code>lastlog</code> , <code>login</code> , <code>logoutd</code> , <code>newgrp</code> , <code>newusers</code> , <code>nologin</code> , <code>passwd</code> , <code>pwck</code> , <code>pwconv</code> , <code>pwunconv</code> , <code>sg</code> (link to <code>newgrp</code>), <code>su</code> , <code>useradd</code> , <code>userdel</code> , <code>usermod</code> , <code>vigr</code> (link to <code>vipw</code>), and <code>vipw</code>
Installed directory:	<code>/etc/default</code>

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
chgrp	Used to update group passwords in batch mode
chpasswd	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
gpasswd	Is used to add and delete members and administrators to groups

groupadd	Creates a group with the given name
groupdel	Deletes the group with the given name
groupmems	Allows a user to administer his/her own group membership list without the requirement of 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. Designed to be used as the default shell for accounts that have been disabled
passwd	Is used to change the password for a user or group account
pwck	Verifies the integrity of the password files <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

10.24. Coreutils-8.21

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

10.24.1. 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-8.21-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
```

The meaning of the configure options:

```
FORCE_UNSAFE_CONFIGURE=1
```

Forces Coreutils to compile when using the root user.

Compile the package:

```
make
```

The test suite of Coreutils makes several assumptions about the presence of system users and groups that are not valid within the minimal environment that exists at the moment. Therefore, additional items need to be set up before running the tests. Skip down to “Install the package” if not running the test suite.

Create two dummy groups and a dummy user:

```
echo "dummy1:x:1000:" >> /etc/group
echo "dummy2:x:1001:dummy" >> /etc/group
echo "dummy:x:1000:1000::/root:/bin/bash" >> /etc/passwd
```

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=dummy SUBDIRS= check-root
```

The testsuite will now be run as the dummy user. Fix the permissions for a few files to allow this:

```
chown -Rv dummy .
```

Then run the remainder of the tests as the dummy user:

```
su dummy -s /bin/bash \
-c "PATH=$PATH make RUN_EXPENSIVE_TESTS=yes -k check || true"
```

When testing is complete, remove the dummy user and groups:

```
sed -i '/dummy/d' /etc/passwd /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,stty,true,uname} /bin
mv -v /usr/bin/chroot /usr/sbin
```

Other Coreutils programs are used by some of the scripts in the CLFS-Bootscripts package. As `/usr` may not be available during the early stages of booting, those binaries need to be on the root partition:

```
mv -v /usr/bin/{[,basename,head,install,nice} /bin
mv -v /usr/bin/{readlink,sleep,sync,test,touch} /bin
ln -svf ../../bin/install /usr/bin
```

10.24.2. 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, od, paste, pathchk, pinky, pr, printenv, printf, ptx, pwd, readlink, realpath, rm, rmdir, runcon, seq, sha1sum, sha224sum, sha256sum, sha384sum, sha512sum, shred, shuf, sleep, sort, split, stat, stdbuf, stty, sum, sync, tac, tail, tee, test, timeout, touch, tr, true, truncate, tsort, tty, uname, unexpand, uniq, unlink, users, vdir, wc, who, whoami, and yes`

Installed library: `libstdbuf.so`

Installed directory: `/usr/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 <code>LS_COLOR</code> 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
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 [test	Compares values and checks file types
timeout	Runs a command with a time limit
touch	Changes file timestamps, setting the access and modification times of the given files to the current time; files that do not exist are created with zero length
tr	Translates, squeezes, and deletes the given characters from standard input
true	Does nothing, successfully; it always exits with a status code indicating success
truncate	Shrinks or expands a file to the specified size
tsort	Performs a topological sort; it writes a completely ordered list according to the partial ordering in a given file
tty	Reports the file name of the terminal connected to standard input
uname	Reports system information
unexpand	Converts spaces to tabs
uniq	Discards all but one of successive identical lines
unlink	Removes the given file
users	Reports the names of the users currently logged on
vdir	Is the same as ls -l
wc	Reports the number of lines, words, and bytes for each given file, as well as a total line when more than one file is given
who	Reports who is logged on
whoami	Reports the user name associated with the current effective user ID
yes	Repeatedly outputs “y” or a given string until killed
libstdbuf	Library used by stdbuf

10.25. Iana-Etc-2.30

The Iana-Etc package provides data for network services and protocols.

10.25.1. Installation of Iana-Etc



Note

This package has the option of downloading updated data when internet access is available. If `/etc/resolv.conf` has a nameserver entry and internet access is available at this step, then apply the IANA get patch and get the updated data:

```
patch -Np1 -i ../iana-etc-2.30-get_fix-1.patch
make get
```

Do not apply the following patch.

The following patch updates the services and protocol files:

```
patch -Np1 -i ../iana-etc-2.30-numbers_update-20120610-2.patch
```

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
```

10.25.2. Contents of Iana-Etc

Installed files: `/etc/protocols` and `/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

10.26. M4-1.4.17

The M4 package contains a macro processor.

10.26.1. 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
```

10.26.2. 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.

10.27. Bison-3.0

The Bison package contains a parser generator.

10.27.1. Installation of Bison

The `configure` script does not determine the correct value for the following. Set the value manually:

```
echo "ac_cv_prog_lex_is_flex=yes" > config.cache
```

Prepare Bison for compilation:

```
./configure --prefix=/usr --cache-file=config.cache
```

Compile the package:

```
make
```

To test the results, issue: `make check`.

Install the package:

```
make install
```

10.27.2. Contents of Bison

Installed programs:	bison and yacc
Installed library:	liby.a
Installed directory:	/usr/share/bison

Short Descriptions

bison	Generates, from a series of rules, a program for analyzing the structure of text files; Bison is a replacement for Yacc (Yet Another Compiler Compiler)
yacc	A wrapper for bison , meant for programs that still call yacc instead of bison ; it calls bison with the <code>-y</code> option
<code>liby.a</code>	The Yacc library containing implementations of Yacc-compatible <code>yyerror</code> and <code>main</code> functions; this library is normally not very useful, but POSIX requires it

10.28. Libtool-2.4.2

The Libtool package contains the GNU generic library support script. It wraps the complexity of using shared libraries in a consistent, portable interface.

10.28.1. 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
```

10.28.2. Contents of Libtool

Installed programs:	libtool and libtoolize
Installed libraries:	libltdl.[a,so]
Installed directories:	/usr/include/libltdl, /usr/share/libtool

Short Descriptions

libtool	Provides generalized library-building support services
libtoolize	Provides a standard way to add libtool support to a package
libltdl	Hides the various difficulties of dlopening libraries

10.29. Flex-2.5.37

The Flex package contains a utility for generating programs that recognize patterns in text.

10.29.1. Installation of Flex

Prepare Flex for compilation:

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

Compile the package:

```
make
```

To test the results, issue: **make check**.

Install the package:

```
make install
```

There are some packages that expect to find the `lex` library in `/usr/lib`. Create a symlink to account for this:

```
ln -sv libfl.a /usr/lib/libl.a
```

A few programs do not know about **flex** yet and try to run its predecessor, **lex**. To support those programs, create a wrapper script named `lex` that calls `flex` in **lex** emulation mode:

```
cat > /usr/bin/lex << "EOF"
#!/bin/sh
# Begin /usr/bin/lex

exec /usr/bin/flex -l "$@"

# End /usr/bin/lex
EOF
chmod -v 755 /usr/bin/lex
```

10.29.2. Contents of Flex

Installed programs: flex and lex
Installed libraries: libfl.a and libfl_pic.a

Short Descriptions

flex A tool for generating programs that recognize patterns in text; it allows for the versatility to specify the rules for pattern-finding, eradicating the need to develop a specialized program

flex++ Link to **flex** which makes it generate C++ scanner classes

lex A script that runs **flex** in **lex** emulation mode

`libfl.a` The `flex` library

`libfl_pic.a` The `flex` library

10.30. IPRoute2-3.10.0

The IPRoute2 package contains programs for basic and advanced IPV4-based networking.

10.30.1. Installation of IPRoute2

By default, this package builds the **arpd** program, which is dependent on Berkeley DB. Because **arpd** is not a very common requirement on a base Linux system, remove the dependency on Berkeley DB by using the commands below. If the **arpd** binary is needed, instructions for compiling Berkeley DB can be found in CBLFS at http://cblfs.cross-lfs.org/index.php/Berkeley_DB.

```
sed -i '/^TARGETS/s@arpd@g' misc/Makefile
sed -i '/ARPD/d' Makefile
rm -v man/man8/arpd.8
```

Remove unused libnl headers:

```
sed -i '/netlink\\//d' ip/ipl2tp.c
```

Compile the package:

```
make DESTDIR= DOCDIR=/usr/share/doc/iproute2 \
      MANDIR=/usr/share/man
```

The meaning of the make option:

DESTDIR=

This option overrides the default *DESTDIR* of */usr*, so that that the IPRoute2 binaries will be installed into */sbin*. This is the correct location according to the FHS, because some of the IPRoute2 binaries are used by the CLFS-Bootscripts package.

DOCDIR=/usr/share/doc/iproute2 MANDIR=/usr/share/man

The *DESTDIR=/* parameter would cause documentation to be installed into */share/doc* and */share/man*. These options ensure the docs are installed to the correct locations.

This package does not come with a test suite.

Install the package:

```
make DESTDIR= DOCDIR=/usr/share/doc/iproute2 \
      MANDIR=/usr/share/man install
```

10.30.2. Contents of IPRoute2

Installed programs: ctstat (link to lstat), genl, ifcfg, ifstat, ip, lstat, nstat, routef, routel, rtacct, rtmon, rtpr, rtstat (link to lstat), ss, and tc

Installed directories: /etc/iproute2, /lib/tc, /usr/lib/tc, /usr/share/doc/iproute2

Short Descriptions

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
route	A component of ip route . This is for flushing the routing tables
route	A component of ip route . This is for listing the routing tables
rtacct	Displays the contents of <code>/proc/net/rt_acct</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 tc qdisc allows users to setup the queuing discipline tc class allows users to setup classes based on the queuing discipline scheduling tc estimator allows users to estimate the network flow into a network tc filter allows users to setup the QOS/COS packet filtering tc policy allows users to setup the QOS/COS policies

10.31. Perl-5.18.1

The Perl package contains the Practical Extraction and Report Language.

10.31.1. Installation of Perl

By default, Perl's `Compress::Raw::Zlib` module builds and links against its own internal copy of Zlib. The following command will tell it to use the system-installed Zlib:

```
sed -i -e '/^BUILD_ZLIB/s/True/False/' \
    -e '/^INCLUDE/s,\.\/zlib-src,\/usr\/include,' \
    -e '/^LIB/s,\.\/zlib-src,\/usr\/lib,' \
    cpan/Compress-Raw-Zlib/config.in
```



Note

If you are following the boot method you will need to enable the loopback device as well as set a hostname for some of the tests:

```
ip link set lo up
hostname clfs
```

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 testsuite:

```
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" \
    -Dusetthreads -Duseshrplib
```

The meaning of the configure option:

`-Dpager="/bin/less -isR"`

This corrects an error in the way that **perldoc** invokes the **less** program.

`-Dman1dir=/usr/share/man/man1 -Dman3dir=/usr/share/man/man3`

Since Groff is not installed yet, **configure.gnu** thinks that we do not want man pages for Perl. Issuing these parameters overrides this decision.

`-Dusetthreads`

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:

```
make install
```

10.31.2. 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, perl5.18.1 (link to perl), perlbug, perldoc, perlivp, perlthanks (link to perlbug), piconv, pl2pm, pod2html, pod2latex, pod2man, pod2text, pod2usage, podchecker, podselect, prove, psed (link to s2p), pstruct (link to c2ph), ptar, ptardiff, ptargrep, s2p, shasum, splain, xsubpp, and zipdetails

Installed libraries: Several hundred which cannot all be listed here

Installed directory: /usr/lib/perl5

Short Descriptions

a2p Translates awk to Perl

c2ph Dumps C structures as generated from **cc -g -S**

config_data Queries or changes configuration of Perl modules

corelist A commandline frontend to Module::CoreList

cpan 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

perl5.18.1 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 <code>.pl</code> files to Perl5 <code>.pm</code> 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 <code>*roff</code> 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 <code>Test::Harness</code>
psed	A Perl version of the stream editor sed
pstruct	Dumps C structures as generated from <code>cc -g -S</code> 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

10.32. Readline-6.2

The Readline package is a set of libraries that offers command-line editing and history capabilities.

10.32.1. 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-6.2-branch_update-3.patch
```

Prepare Readline for compilation:

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

Compile the package:

```
make SHLIB_LIBS=-lncurses
```

This package does not come with a test suite.

Install the package:

```
make install
```

Install the documentation:

```
make install-doc
```

Now move the static libraries to a more appropriate location:

```
mv -v /lib/lib{readline,history}.a /usr/lib
```

Next, remove the .so files in /lib and relink them into /usr/lib.

```
rm -v /lib/lib{readline,history}.so
ln -svf ../../lib/libreadline.so.6 /usr/lib/libreadline.so
ln -svf ../../lib/libhistory.so.6 /usr/lib/libhistory.so
```

10.32.2. Contents of Readline

Installed libraries:	libhistory.[a,so], and libreadline.[a,so]
Installed directories:	/usr/include/readline, /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

10.33. Autoconf-2.69

The Autoconf package contains programs for producing shell scripts that can automatically configure source code.

10.33.1. 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
```

10.33.2. Contents of Autoconf

Installed programs: autoconf, autoheader, autom4te, autoreconf, autoscan, autoupdate, and ifnames
Installed directory: /usr/share/autoconf

Short Descriptions

autoconf	Produces shell scripts that automatically configure software source code packages to adapt to many kinds of Unix-like systems. The configuration scripts it produces are independent—running them does not require the autoconf program.
autoheader	A tool for creating template files of C <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

10.34. Automake-1.12.4

The Automake package contains programs for generating Makefiles for use with Autoconf.

10.34.1. Installation of Automake

Prepare Automake for compilation:

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

Compile the package:

```
make
```

To test the results, issue: **make check**.

Install the package:

```
make install
```

10.34.2. Contents of Automake

Installed programs: acinstall, aclocal, aclocal-1.12, automake, automake-1.12, compile, config.guess, config.sub, depcomp, elisp-comp, install-sh, mdate-sh, missing, mkinstalldirs, py-compile, symlink-tree, and ylwrap

Installed directories: /usr/share/aclocal-1.12, /usr/share/automake-1.12, /usr/share/doc/automake

Short Descriptions

acinstall	A script that installs aclocal-style M4 files
aclocal	Generates aclocal.m4 files based on the contents of configure.in files
aclocal-1.12	A hard link to aclocal
automake	A tool for automatically generating Makefile.in files from Makefile.am files. To create all the Makefile.in files for a package, run this program in the top-level directory. By scanning the configure.in file, it automatically finds each appropriate Makefile.am file and generates the corresponding Makefile.in file
automake-1.12	A hard link to automake
compile	A wrapper for compilers
config.guess	A script that attempts to guess the canonical triplet for the given build, host, or target architecture
config.sub	A configuration validation subroutine script
depcomp	A script for compiling a program so that dependency information is generated in addition to the desired output
elisp-comp	Byte-compiles Emacs Lisp code
install-sh	A script that installs a program, script, or data file
mdate-sh	A script that prints the modification time of a file or directory
missing	A script acting as a common stub for missing GNU programs during an installation

minstalldirs	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

10.35. Bash-4.2

The Bash package contains the Bourne-Again SHell.

10.35.1. 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-4.2-branch_update-7.patch
```

Prepare Bash for compilation:

```
./configure --prefix=/usr --bindir=/bin \
  --without-bash-malloc --with-installed-readline
```

The meaning of the 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 htmdir=/usr/share/doc/bash-4.2 install
```

Run the newly compiled `bash` program (replacing the one that is currently being executed):

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



Note

The parameters used make the `bash` process an interactive login shell and continue to disable hashing so that new programs are found as they become available.

10.35.2. Contents of Bash

Installed programs: bash, bashbug, and sh (link to bash)
Installed directory: /usr/share/doc/bash-4.2

Short Descriptions

bash A widely-used command interpreter; it performs many types of expansions and substitutions on a given command line before executing it, thus making this interpreter a powerful tool

bashbug A shell script to help the user compose and mail standard formatted bug reports concerning **bash**

sh A symlink to the **bash** program; when invoked as **sh**, **bash** tries to mimic the startup behavior of historical versions of **sh** as closely as possible, while conforming to the POSIX standard as well

10.36. Bc-1.06.95

The Bc package contains an arbitrary precision numeric processing language.

10.36.1. Installation of Bc

Prepare Bc for compilation:

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

Compile the package:

```
make
```

To test the results, issue: `echo "quit" | ./bc/bc -l Test/checklib.b`

Install the package:

```
make install
```

10.36.2. Contents of Bc

Installed programs: bc and dc

Short Descriptions

bc is a command line calculator

dc is a reverse-polish command line calculator

10.37. 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**.

10.37.1. 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
```

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
```

10.37.2. 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 bzmores), and bzmores

Installed libraries: libbz2.a, libbz2.so (link to libbz2.so.1.0), libbz2.so.1.0 (link to libbz2.so.1.0.6), and 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
<code>libbz2*</code>	The library implementing lossless, block-sorting data compression, using the Burrows-Wheeler algorithm

10.38. Diffutils-3.3

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

10.38.1. Installation of Diffutils

Prepare Diffutils for compilation:

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

Diffutils wants **ed** as the default editor. The following sed will change the default to **vim**:

```
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
```

10.38.2. Contents of Diffutils

Installed programs: cmp, diff, diff3, and sdiff

Short Descriptions

- cmp** Compares two files and reports whether or in which bytes they differ
- diff** Compares two files or directories and reports which lines in the files differ
- diff3** Compares three files line by line
- sdiff** Merges two files and interactively outputs the results

10.39. File-5.15

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

10.39.1. Installation of File

Prepare File for compilation:

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

Compile the package:

```
make
```

This package does not come with a test suite.

Install the package:

```
make install
```

10.39.2. Contents of File

Installed programs:	file
Installed library:	libmagic.[a,so]

Short Descriptions

file	Tries to classify each given file; it does this by performing several tests—file system tests, magic number tests, and language tests
libmagic	Contains routines for magic number recognition, used by the file program

10.40. Gawk-4.1.0

The Gawk package contains programs for manipulating text files.

10.40.1. 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
```

10.40.2. Contents of Gawk

Installed programs: awk (link to gawk), gawk, gawk-4.1.0, grcat, igawk, pgawk, pgawk-4.1.0, and pwcats
Installed directories: /usr/lib/awk, /usr/share/awk

Short Descriptions

awk	A link to gawk
gawk	A program for manipulating text files; it is the GNU implementation of awk
gawk-4.1.0	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-4.1.0	Hard link to pgawk
pwcats	Dumps the password database <code>/etc/passwd</code>

10.41. Findutils-4.4.2

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

10.41.1. 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
```

The **find** program is used by some of the scripts in the CLFS-Bootscripts package. As `/usr` may not be available during the early stages of booting, the **find** binary needs to be on the root partition:

```
mv -v /usr/bin/find /bin
```

The **updatedb** script needs to be modified to point to the new location for **find**:

```
sed -i 's@find:=${BINDIR}@find:="/bin@' /usr/bin/updatedb
```

10.41.2. Contents of Findutils

Installed programs: bigram, code, find, frcode, locate, oldfind, updatedb, and 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

10.42. Gettext-0.18.3.1

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.

10.42.1. Installation of Gettext

Prepare Gettext 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 Gettext

Installed programs:	autopoint, config.charset, config.rpath, envsubst, gettext, gettext.sh, gettextize, hostname, msgattrib, msgcat, msgcmp, msgcomm, msgconv, msgen, msgexec, msgfilter, msgfmt, msggrep, msginit, msgmerge, msgunfmt, msguniq, ngettext, recode-sr-latin, and xgettext
Installed libraries:	libasprintf.[a,so], libgettextlib.so, libgettextpo.[a,so], libgettextsrc.so, and preloadable_libintl.so
Installed directories:	/usr/lib/gettext, /usr/share/doc/gettext, /usr/share/gettext

Short Descriptions

autopoint	Copies standard Gettext infrastructure files into a source package
config.charset	Outputs a system-dependent table of character encoding aliases
config.rpath	Outputs a system-dependent set of variables, describing how to set the runtime search path of shared libraries in an executable
envsubst	Substitutes environment variables in shell format strings
gettext	Translates a natural language message into the user's language by looking up the translation in a message catalog
gettext.sh	Primarily serves as a shell function library for gettext
gettextize	Copies all standard Gettext files into the given top-level directory of a package to begin internationalizing it
hostname	Displays a network hostname in various forms
msgattrib	Filters the messages of a translation catalog according to their attributes and manipulates the attributes
msgcat	Concatenates and merges the given .po files
msgcmp	Compares two .po files to check that both contain the same set of msgid strings

msgcomm	Finds the messages that are common to to the given .po files
msgconv	Converts a translation catalog to a different character encoding
mngen	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
<code>libasprintf</code>	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
<code>libgettextlib</code>	a private library containing common routines used by the various Gettext programs; these are not intended for general use
<code>libgettextpo</code>	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 mngen) will not suffice
<code>libgettextsrc</code>	A private library containing common routines used by the various Gettext programs; these are not intended for general use
<code>preloadable_libintl.so</code>	A library, intended to be used by LD_PRELOAD, that assists libintl in logging untranslated messages.

10.43. Grep-2.14

The Grep package contains programs for searching through files.

10.43.1. 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
```

10.43.2. Contents of Grep

Installed programs: egrep, fgrep, and grep

Short Descriptions

egrep Prints lines matching an extended regular expression
fgrep Prints lines matching a list of fixed strings
grep Prints lines matching a basic regular expression

10.44. Groff-1.22.2

The Groff package contains programs for processing and formatting text.

10.44.1. 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
```

Some documentation programs, such as **xman**, will not work properly without the following symlinks:

```
ln -sv soelim /usr/bin/zsoelim
ln -sv eqn /usr/bin/geqn
ln -sv tbl /usr/bin/gtbl
```

10.44.2. Contents of Groff

Installed programs: addftinfo, afmtodit, chem, eqn, eqn2graph, gdiffmk, geqn (link to eqn), grap2graph, grn, grodvi, groff, groffer, grog, grolbp, grolj4, grops, grotty, gtbl (link to tbl), hpftodit, indxbib, lkbib, lookbib, mmroff, neqn, nroff, pdfroff, pfbtops, pic, pic2graph, post-grohtml, pre-grohtml, preconv, refer, roff2dvi, roff2html, roff2pdf, roff2ps, roff2text, roff2x, soelim, tbl, tfmtodit, troff, and zsoelim (link to soelim)

Installed directories: /usr/lib/groff, /usr/share/doc/groff-1.22.2, /usr/share/groff

Short Descriptions

addftinfo	Reads a troff font file and adds some additional font-metric information that is used by the groff system
afmtodit	Creates a font file for use with groff and grops
chem	Groff preprocessor for producing chemical structure diagrams
eqn	Compiles descriptions of equations embedded within troff input files into commands that are understood by troff
eqn2graph	Converts a troff EQN (equation) into a cropped image
gdiffmk	Marks differences between groff/nroff/troff files
geqn	A link to eqn
grap2graph	Converts a grap diagram into a cropped bitmap image

grn	A groff preprocessor for gremlin files
grodvi	A driver for groff that produces TeX dvi format
groff	A front-end to the groff document formatting system; normally, it runs the troff program and a post-processor appropriate for the selected device
groffer	Displays groff files and man pages on X and tty terminals
grog	Reads files and guesses which of the groff options <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
gtbl	A link to tbl
hpftodit	Creates a font file for use with groff -Tlj4 from an HP-tagged font metric file
indxbib	Creates an inverted index for the bibliographic databases with a specified file for use with refer , lookbib , and lkbib
lkbib	Searches bibliographic databases for references that contain specified keys and reports any references found
lookbib	Prints a prompt on the standard error (unless the standard input is not a terminal), reads a line containing a set of keywords from the standard input, searches the bibliographic databases in a specified file for references containing those keywords, prints any references found on the standard output, and repeats this process until the end of input
mmroff	A simple preprocessor for groff
neqn	Formats equations for American Standard Code for Information Interchange (ASCII) output
nroff	A script that emulates the nroff command using groff
pdfroff	Creates pdf documents using groff
pfbtops	Translates a PostScript font in <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 <i>.so file</i> 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
tfmto dit	Creates a font file for use with groff -Tdvi
troff	Is highly compatible with Unix troff ; it should usually be invoked using the groff command, which will also run preprocessors and post-processors in the appropriate order and with the appropriate options
zsoelim	A link to soelim

10.45. Less-460

The Less package contains a text file viewer.

10.45.1. Installation of Less

Prepare Less for compilation:

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

The meaning of the configure option:

```
--sysconfdir=/etc
```

This option tells the programs created by the package to look in `/etc` for the configuration files.

Compile the package:

```
make
```

This package does not come with a test suite.

Install the package:

```
make install
```

Move `less` to `/bin`:

```
mv -v /usr/bin/less /bin
```

10.45.2. Contents of Less

Installed programs: `less`, `lessecho`, and `lesskey`

Short Descriptions

less	A file viewer or pager; it displays the contents of the given file, letting the user scroll, find strings, and jump to marks
lessecho	Needed to expand meta-characters, such as <code>*</code> and <code>?</code> , in filenames on Unix systems
lesskey	Used to specify the key bindings for less

10.46. Gzip-1.6

The Gzip package contains programs for compressing and decompressing files.

10.46.1. 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/z{egrep,cmp,diff,fgrep,force,grep,less,more,new} /usr/bin
```

10.46.2. Contents of Gzip

Installed programs: gunzip, gzexe, gzip, uncompress, zcat, zcmp, zdiff, zegrep, zfgrep, zforce, zgrep, zless, zmore, and znew

Short Descriptions

gunzip	Decompresses gzipped files
gzexe	Creates self-decompressing executable files
gzip	Compresses the given files using Lempel-Ziv (LZ77) coding
uncompress	Decompresses compressed files
zcat	Decompresses the given gzipped files to standard output
zcmp	Runs cmp on gzipped files
zdiff	Runs diff on gzipped files
zegrep	Runs egrep on gzipped files
zfgrep	Runs fgrep on gzipped files
zforce	Forces a <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— <code>.Z</code> to <code>.gz</code>

10.47. IPutils-s20121221

The IPutils package contains programs for basic networking.

10.47.1. Installation of IPutils

IPutils has various issues addressed by the following patch:

```
patch -Np1 -i ../iputils-s20121221-fixes-1.patch
```

Compile the package:

```
make USE_CAP=no \  
    IPV4_TARGETS="tracepath ping clockdiff rdisc" \  
    IPV6_TARGETS="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
```

10.47.2. Contents of iputils

Installed programs: clockdiff, ping, rdisc, tracepath, tracepath6, and traceroute6

Short Descriptions

clockdiff	Measures the clock difference between hosts
ping	Sends echo-request packets and reports how long the replies take. This is the IPV4 version
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

10.48. Kbd-2.0.0

The Kbd package contains key-table files and keyboard utilities.

10.48.1. Installation of Kbd

Prepare Kbd for compilation:

```
PKG_CONFIG_PATH="/tools/lib/pkgconfig" \
./configure --prefix=/usr --disable-vlock --enable-optional-progs
```

Compile the package:

```
make
```

This package does not come with a test suite.

Install the package:

```
make install
```

Some of the programs from Kbd are used by scripts in the CLFS-Bootscripts package. As `/usr` may not be available during the early stages of booting, those binaries need to be on the root partition:

```
mv -v /usr/bin/{kbd_mode,dumpkeys,loadkeys,openvt,setfont,setvtrgb} /bin
```

10.48.2. 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), psfstrietable (link to psfxtable), psfxtable, resizecons, setfont, setkeycodes, setleds, setmetamode, setvtrgb, showconsolefont, showkey, unicode_start, and unicode_stop
Installed directories:	/usr/share/consolefonts, /usr/share/consoletrans, /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
kbinfo	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	A link to psfxtable
psfgettable	A link to psfxtable
psfstriptime	A link to psfxtable
psfxtable	Handle Unicode character tables for console fonts
resizecons	Changes the kernel idea of the console size
setfont	Changes the Enhanced Graphic Adapter (EGA) and Video Graphics Array (VGA) fonts on the console
setkeycodes	Loads kernel scancode-to-keycode mapping table entries; this is useful if there are unusual keys on the keyboard
setleds	Sets the keyboard flags and Light Emitting Diodes (LEDs)
setmetamode	Defines the keyboard meta-key handling
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

10.49. Make-3.82

The Make package contains a program for compiling packages.

10.49.1. Installation of Make

Apply upstream fixes:

```
patch -Np1 -i ../make-3.82-fixes-1.patch
```

Prepare Make for compilation:

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

Compile the package:

```
make
```

To test the results, issue: **make check**.

Install the package:

```
make install
```

10.49.2. Contents of Make

Installed program: **make**

Short Descriptions

make Automatically determines which pieces of a package need to be (re)compiled and then issues the relevant commands

10.50. XZ-Utills-5.0.5

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

10.50.1. Installation of XZ-Utills

Prepare XZ-Utills for compilation:

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

Compile the package:

```
make
```

To test the results, issue: **make check**.

Install the programs:

```
make pkgconfigdir=/usr/lib/pkgconfig 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
```

10.50.2. Contents of XZ-Utills

Installed programs:	lzcat (link to xz), lzcmp (link to lzdiff), lzdiff, lzgrep (link to lzgrep), lzfgrep (link to lzgrep), lzgrep, lzless (link to lzmore), lzma (link to xz), lzmadec, lzmore, unlzma (link to xz), unxz (link to xz), xz, xzcat (link to xz), and xzdec
Installed libraries:	liblzma.[a,so]
Installed directories:	/usr/include/lzma, /usr/share/doc/xz

Short Descriptions

lzcat	Decompresses LZMA and xz files
lzcmp	Compares lzma compressed files
lzdiff	Compares lzma compressed files
lzgrep	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
lzmore	Runs more on lzma files
unlzma	Uncompresses lzma files
unxz	Uncompresses xz files
xz	Creates xz compressed files

xzcat Decompresses xz files
xzdec Decompresses to standard output
liblzma The LZMA library

10.51. Man-1.6g

The Man package contains programs for finding and viewing man pages.

10.51.1. Installation of Man

This patch adds support for Internationalization:

```
patch -Np1 -i ../man-1.6g-i18n-1.patch
```

A few adjustments need to be made to the sources of Man.

First, a **sed** substitution is needed to add the `-R` switch to the `PAGER` variable so that escape sequences are properly handled by Less:

```
sed -i 's@-is@&R@g' configure
```

Another couple of **sed** substitutions comment out the “`MANPATH /usr/man`” and “`MANPATH /usr/local/man`” lines in the `man.conf` file to prevent redundant results when using programs such as **whatis**:

```
sed -i 's@MANPATH./usr/man@#&@g' src/man.conf.in
sed -i 's@MANPATH./usr/local/man@#&@g' src/man.conf.in
```

Prepare Man for compilation:

```
./configure -confdir=/etc
```

The meaning of the configure options:

`-confdir=/etc`

This tells the **man** program to look for the `man.conf` configuration file in the `/etc` directory.

Compile the package:

```
make
```

This package does not come with a test suite.

Install the package:

```
make install
```



Note

If you will be working on a terminal that does not support text attributes such as color and bold, you can disable Select Graphic Rendition (SGR) escape sequences by editing the `man.conf` file and adding the `-c` option to the `NROFF` variable. If you use multiple terminal types for one computer it may be better to selectively add the `GROFF_NO_SGR` environment variable for the terminals that do not support SGR.

If the character set of the locale uses 8-bit characters, search for the line beginning with “`NROFF`” in `/etc/man.conf`, and verify that it matches the following:

```
NROFF /usr/bin/nroff -Tlatin1 -mandoc
```

Note that “latin1” should be used even if it is not the character set of the locale. The reason is that, according to the specification, **groff** has no means of typesetting characters outside International Organization for Standards (ISO) 8859-1 without some strange escape codes. When formatting man pages, **groff** thinks that they are in the ISO 8859-1 encoding and this `-Tlatin1` switch tells **groff** to use the same encoding for output. Since **groff** does no recoding of input characters, the formatted result is really in the same encoding as input, and therefore it is usable as the input for a pager.

This does not solve the problem of a non-working **man2dvi** program for localized man pages in non-ISO 8859-1 locales. Also, it does not work with multibyte character sets. The first problem does not currently have a solution. The second issue is not of concern because the CLFS installation does not support multibyte character sets.

10.51.2. Contents of Man

Installed programs: apropos, makewhatis, man, man2dvi, man2html, and whatis

Short Descriptions

apropos	Searches the whatis database and displays the short descriptions of system commands that contain a given string
makewhatis	Builds the whatis database; it reads all the man pages in the MANPATH and writes the name and a short description in the whatis database for each page
man	Formats and displays the requested on-line man page
man2dvi	Converts a man page into dvi format
man2html	Converts a man page into HTML
whatis	Searches the whatis database and displays the short descriptions of system commands that contain the given keyword as a separate word

10.52. Kmod-15

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

10.52.1. Installation of Kmod

Prepare Kmod for compilation:

```
./configure --prefix=/usr \
  --bindir=/bin --sysconfdir=/etc \
  --with-rootlibdir=/lib --disable-manpages \
  --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
make -C man install
```

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

```
ln -sfv kmod /bin/lsmmod
ln -sfv ../bin/kmod /sbin/depmod
ln -sfv ../bin/kmod /sbin/inssmod
ln -sfv ../bin/kmod /sbin/modprobe
ln -sfv ../bin/kmod /sbin/modinfo
ln -sfv ../bin/kmod /sbin/rmmmod
```

10.52.2. Contents of Kmod

Installed programs: depmod, inssmod, kmod, lsmmod, modinfo, modprobe, and rmmmod

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

inssmod 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
rmmmod	Unloads modules from the running kernel

10.53. Patch-2.7.1

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

10.53.1. 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
```

10.53.2. Contents of Patch

Installed program: patch

Short Descriptions

patch Modifies files according to a patch file. A patch file is normally a difference listing created with the **diff** program. By applying these differences to the original files, **patch** creates the patched versions.

10.54. Psmisc-22.20

The Psmisc package contains programs for displaying information about running processes.

10.54.1. Installation of Psmisc

Prepare Psmisc for compilation:

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

The meaning of the configure option:

```
--exec-prefix=""
```

This ensures that the Psmisc binaries will install into `/bin` instead of `/usr/bin`. This is the correct location according to the FHS, because some of the Psmisc binaries are used by the CLFS-Bootscripts package.

Compile the package:

```
make
```

This package does not come with a test suite.

Install the package:

```
make install
```

There is no reason for the `pstree` and `pstree.x11` programs to reside in `/bin`. Therefore, move them to `/usr/bin`:

```
mv -v /bin/pstree* /usr/bin
```

By default, Psmisc's `pidof` program is not installed. This usually is not a problem because it is installed later in the Sysvinit package, which provides a better `pidof` program. If Sysvinit will not be used for a particular system, complete the installation of Psmisc by creating the following symlink:

```
ln -sv killall /bin/pidof
```

10.54.2. Contents of Psmisc

Installed programs: `fuser`, `killall`, `peekfd`, `prtstat`, `pstree`, and `pstree.x11` (link to `pstree`)

Short Descriptions

fuser	Reports the Process IDs (PIDs) of processes that use the given files or file systems
killall	Kills processes by name; it sends a signal to all processes running any of the given commands
peekfd	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

10.55. Libestr-0.1.5

The Libestr package is a library for some string essentials.

10.55.1. Installation of Libestr

Prepare Libestr for compilation:

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

Compile the package:

```
make
```

This package does not come with a test suite.

Install the package:

```
make install
```

10.55.2. Contents of Libestr

Installed libraries: libestr.[a,so]

Short Descriptions

`libestr` contains functions for aiding in string functions

10.56. Libee-0.4.1

The Libee is an event expression library.

10.56.1. Installation of Libee

Prepare Libee for compilation:

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

Compile the package:



Note

Libee will fail to compile if using multiple jobs with make. Append "-j 1" to the following make command:

```
make
```

This package does not come with a test suite.

Install the package:

```
make install
```

10.56.2. Contents of Libee

Installed Program:	libee-convert
Installed libraries:	libee.[a,so]
Installed directory:	/usr/include/libee

Short Descriptions

libee-convert	todo
libee	is the event expression library

10.57. Rsyslog-6.4.2

The rsyslog package contains programs for logging system messages, such as those given by the kernel when unusual things happen.

10.57.1. Installation of Rsyslog

Prepare Rsyslog for compilation:

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

Compile the package:

```
make
```

To test the results, issue: **make check**.

Install the package:

```
make install
```

Create a directory for expansion snippets:

```
install -dv /etc/rsyslog.d
```

```

$ModLoad imklog.so

#####
# Global Options
# Use traditional timestamp format.
$ActionFileDefaultTemplate RSYSLOG_TraditionalFileFormat

# Set the default permissions for all log files.
$FileOwner root
$FileGroup root
$FileCreateMode 0640
$DirCreateMode 0755

# Provides UDP reception
$ModLoad imudp
$UDPServerRun 514

# Disable Repeating of Entries
$RepeatedMsgReduction on

#####
# Include Rsyslog Config Snippets

$IncludeConfig /etc/rsyslog.d/*.conf

#####
# Standard Log Files

auth,authpriv.*    /var/log/auth.log
*.*;auth,authpriv.none  -/var/log/syslog
daemon.*           -/var/log/daemon.log
kern.*             -/var/log/kern.log
lpr.*              -/var/log/lpr.log
mail.*             -/var/log/mail.log
user.*            -/var/log/user.log

# Catch All Logs
*.=debug;\
auth,authpriv.none;\
news.none;mail.none -/var/log/debug
*.=info;*.=notice;*.=warn;\
auth,authpriv.none;\
cron,daemon.none;\
mail,news.none -/var/log/messages

# Emergencies are shown to everyone
*.emerg           *

# End /etc/rsyslog.conf
EOF

```


10.57.3. Contents of rsyslog

Installed programs: rsyslogd
Installed directory: /usr/lib/rsyslog

Short Descriptions

rsyslogd Logs the messages that system programs offer for logging. Every logged message contains at least a date stamp and a hostname, and normally the program's name too, but that depends on how trusting the logging daemon is told to be.

10.58. Sysvinit-2.88dsf

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

10.58.1. Installation of Sysvinit

Apply a sed which disables sulogin, mountpoint, wall, and utmpdump from being built and installed as they are provided by Util-linux:

```
sed -i -e 's/\ sulogin[^\ ]*//' \
    -e '/utmpdump/d' -e '/mountpoint/d' src/Makefile
```

Compile the package:

```
make -C src clobber
make -C src
```

Install the package:

```
make -C src install
```

10.58.2. Configuring Sysvinit

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

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

id:3:initdefault:

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

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

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

su:S016:once:/sbin/sulogin

EOF
```

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

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

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

EOF
```

Finally, add the end line to `/etc/inittab`:

```
cat >> /etc/inittab << "EOF"
# End /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. The **console** initscript from the CLFS-Bootscripts package calls the **setfont** program during system startup. Sending this escape sequence is necessary for people who use non-ISO 8859-1 screen fonts, but it does not affect native English speakers.

10.58.3. Contents of Sysvinit

Installed programs: bootlogd, fstab-decode, halt, init, killall5, last, lastb (link to last), mesg, pidof (link to killall5), poweroff (link to halt), reboot (link to halt), runlevel, shutdown, 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

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>
mesg	Controls whether other users can send messages to the current user's terminal
pidof	Reports the PIDs of the given programs
poweroff	Tells the kernel to halt the system and switch off the computer (see halt)
reboot	Tells the kernel to reboot the system (see halt)
runlevel	Reports the previous and the current run-level, as noted in the last run-level record in <code>/var/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

10.59. Tar-1.26

The Tar package contains an archiving program.

10.59.1. Installation of Tar

The following patch adds a man page for **tar**:

```
patch -Np1 -i ../tar-1.26-man-1.patch
```

EGLIBC-2.18 does not declare gets():

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

Prepare Tar for compilation:

```
FORCE_UNSAFE_CONFIGURE=1 ./configure --prefix=/usr \  
--bindir=/bin --libexecdir=/usr/sbin
```

Compile the package:

```
make
```

To test the results, issue: **make check**.

Install the package:

```
make install
```

10.59.2. Contents of Tar

Installed programs: rmt and tar

Short Descriptions

rmt Remotely manipulates a magnetic tape drive through an interprocess communication connection

tar Creates, extracts files from, and lists the contents of archives, also known as tarballs

10.60. Texinfo-4.13a

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

10.60.1. Installation of Texinfo

The following patch will add support for new compressors like XZ Utils:

```
patch -Np1 -i ../texinfo-4.13a-new_compressors-1.patch
```

Prepare Texinfo for compilation:

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

Compile the package:

```
make
```

To test the results, issue: **make check**.

Install the package:

```
make install
```

The Info documentation system uses a plain text file to hold its list of menu entries. The file is located at `/usr/share/info/dir`. Unfortunately, due to occasional problems in the Makefiles of various packages, it can sometimes get out of sync with the info pages installed on the system. If the `/usr/share/info/dir` file ever needs to be recreated, the following optional commands will accomplish the task:

```
pushd /usr/share/info
rm dir
for f in *
do install-info $f dir 2>/dev/null
done
popd
```

10.60.2. Contents of Texinfo

Installed programs: info, infokey, install-info, makeinfo, pdftexi2dvi, texi2dvi, texi2pdf, and texindex
Installed directory: /usr/share/texinfo

Short Descriptions

info Used to read info pages which are similar to man pages, but often go much deeper than just explaining all the 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

10.61. Eudev-1.3

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

10.61.1. Installation of Eudev

Prepare Eudev for compilation:

```
./configure --prefix=/usr --sysconfdir=/etc \
  --with-rootprefix="" --libexecdir=/lib --enable-split-usr \
  --libdir=/usr/lib --with-rootlibdir=/lib --sbindir=/sbin --bindir=/sbin \
  --enable-rule_generator --disable-introspection --disable-keymap \
  --disable-gudev --disable-gtk-doc-html --with-firmware-path=/lib/firmware \
  --enable-libkmod
```

Compile the package:

```
make
```

To test the results, issue: **make check**.

Install the package:

```
make install
```

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

```
install -dv /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*" \
> /etc/udev/rules.d/80-net-name-slot.rules
```

10.61.2. 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:	/etc/udev, /lib/firmware, /lib/udev, /usr/share/doc/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.
<code>libudev</code>	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.

10.62. Vim-7.4

The Vim package contains a powerful text editor.

10.62.1. Installation of Vim



Alternatives to Vim

If you prefer another editor—such as Emacs, Joe, or Nano—please refer to http://cblfs.cross-lfs.org/index.php/Category:Text_Editors for suggested installation instructions.

The following patch merges all updates from the 7.4 Branch from the Vim developers:

```
patch -Np1 -i ../vim-7.4-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 --enable-multibyte
```

The meaning of the configure options:

--enable-multibyte

This optional but highly recommended switch enables support for editing files in multibyte character encodings. This is needed if using a locale with a multibyte character set. This switch is also helpful to be able to edit text files initially created in Linux distributions like Fedora that use UTF-8 as a default character set.

Compile the package:

```
make
```

To test the results, issue: **make test**. However, this test suite outputs a lot of binary data to the screen, which can cause issues with the settings of the current terminal. This can be resolved by redirecting the output to a log file.

Install the package:

```
make install
```

Many users are 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-7.4`, making it consistent with the location of documentation for other packages:

```
ln -sv ../vim/vim74/doc /usr/share/doc/vim-7.4
```

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.cross-lfs.org/index.php/Vim>.

10.62.2. Configuring Vim

By default, **vim** runs in vi-incompatible mode. This may be new to users who have used other editors in the past. The “*nocompatible*” setting is included below to highlight the fact that a new behavior is being used. It also reminds those who would change to “*compatible*” mode that it should be the first setting in the configuration file. This is necessary because it changes other settings, and overrides must come after this setting. Create a default **vim** configuration file by running the following:

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

set nocompatible
set backspace=2
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'
```

10.62.3. 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, vim132, vim2html.pl, vimdiff (link to vim), vimmm, vimspell.sh, vimtutor, and 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 Hypertext 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

10.63. Hfsutils-3.2.6

The Hfsutils package contains a number of utilities for accessing files on hfs filesystems. It is needed to run **ybin**.

10.63.1. Installation of Hfsutils

The following patch contains fixes. One that fixes a missing errno.h, the other that allows HFSutils to recognize beyond 2gb devices:

```
patch -Np1 -i ../hfsutils-3.2.6-fixes-1.patch
```

Prepare Hfsutils for compilation:

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

Compile the package:

```
make
```

Install the package:

```
make install
```

10.63.2. Contents of Hfsutils

Installed programs: hattrib, hcd, hcopy, hdel, hdir, hfsutils, hformat, hls, hmkdir, hmount, hpwd, hrename, hrmdir, humount, hvol (these are all hardlinks to hfsutils).

Short Descriptions

hattrib	Change FS file or directory attributes.
hcd	Change working HFS directory.
hcopy	Copy files to or from an HFS volume.
hdel	Delete both forks of an HFS file.
hdir	Display an HFS directory in long format.
hformat	Create a new HFS filesystem and make it current.
hfsutils	Tools for accessing Macintosh HFS-formatted volumes.
hls	List files in an HFS directory.
hmkdir	Create a new HFS directory.
hmount	Introduce a new HFS volume and make it current.
hpwd	Print the full path to the current HFS working directory.
hrename	Rename or move an HFS file or directory.
hrmdir	Remove an empty HFS directory.
humount	Remove an HFS volume from the list of known volumes.
hvol	Display or change the current HFS volume.

10.64. Parted-3.1

Parted is a program for creating, copying and modifying partitions, and the file systems on them. Parted is especially useful on PPC machines in that, unlike **fdisk**, it accurately reads Macintosh partition maps.

10.64.1. Installation of Parted

Prepare Parted for compilation:

```
./configure --prefix=/usr --disable-device-mapper
```

The meaning of the configure options:

--disable-device-mapper

This disables the use of the device-mapper library, which we do not install in CLFS.

Compile the Parted package:

```
make
```

Install the package:

```
make install
```

10.64.2. Contents of Parted

Installed programs: parted, partprobe
Installed libraries: libparted.[a,so]

Short Descriptions

parted A program for creating, destroying, resizing, checking and copying partitions, and the filesystems on them. This is useful for creating space for new operating systems, reorganising disk usage, copying data between hard disks, and disk imaging.

partprobe Informs the OS of partition table changes.

libparted A library to manipulate partitions.

10.65. Powerpc-Utills_1.1.3

The Powerpc-Utills package contains a number of utilities for Power Macintoshes and other similar machines. Most of these utilities are now obsolete, but `nvsetenv` is needed by `ybin` to install the bootloader on an hfs partition.

10.65.1. Installation of Powerpc-Utills

This package, originally `pmac-utils`, has issues with NewWorld Macintoshes. The following patch fixes these issues and generally updates the package:

```
patch -Np1 -i ../powerpc-utills_1.1.3-fixes-2.patch
```

Compile the needed programs:

```
make nvsetenv nvsetvol
```

Install the package:

```
install -v -m755 nvsetenv nvsetvol /usr/sbin
install -v -m644 nvsetenv.8 nvsetvol.8 /usr/share/man/man8
```

10.65.2. Contents of Powerpc-Utills

Installed programs: `nvsetenv`, `nvsetvol`

Short Descriptions

`nvsetenv` Manipulate variables in the non-volatile RAM.
`nvsetvol` Adjust the volume of the boot-up chime on Macintoshes.

10.66. Yaboot-1.3.17

The Yaboot package contains a PowerPC Boot Loader for machines using Open Firmware such as NewWorld Macintoshes.

10.66.1. Installation of Yaboot

The following patch adds stub functions for newer e2fsprogs releases.

```
patch -Np1 -i ../yaboot-1.3.17-stubfuncs-1.patch
```

The supplied man pages have `/usr/local` in the text. This sed will correct that:

```
sed -i 's%/usr/local%/usr%' man/*
```

Prevent the build from failing due to warnings with the following sed:

```
sed -i 's%-Werror%%' Makefile
```

Compile the package:

```
make PREFIX=/usr
```

Install the package:

```
make PREFIX=/usr install
```

10.66.2. Contents of Yaboot

Installed programs:	addnote, mkofboot (link to ybin), ofboot, ofpath, yaboot, yabootconfig, ybin
Installed files:	yaboot.conf

Short Descriptions

addnote	For IBM CHRP machines, add a PT_NOTE program header entry to an elf file so that it can be booted.
mkofboot	Format the bootstrap partition and install the yaboot boot loader.
ofboot	Script to format the boot menu using yaboot.conf and write the resulting Open Firmware code to the bootstrap.
ofpath	Determine Open Firmware path corresponding to a device node.
yaboot	Open Firmware boot loader.
yabootconfig	Generate and install a simple yaboot.conf.
ybin	Shell script to update or install the boot loader on a bootstrap partition.
yaboot.conf	Configuration file used by ybin to determine how to install yaboot on the bootstrap partition.

10.67. 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.

10.68. 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. Setting Up System Bootscripts

11.1. Introduction

This chapter details how to install and configure the CLFS-Bootscripts package. Most of these scripts will work without modification, but a few require additional configuration files because they deal with hardware-dependent information.

System-V style init scripts are employed in this book because they are widely used. For additional options, a hint detailing the BSD style init setup is available at <http://hints.cross-lfs.org/index.php/BSD-Init>. Searching the LFS mailing lists for “depinit” will also offer additional choices.

If using an alternative style of init scripts, skip this chapter and move on to Making the CLFS System Bootable.

11.2. Bootscripts for CLFS 2.1-pre1

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

11.2.1. Installation of Bootscripts

Install the package:

```
make install-bootscripts
```

You can will need to run the following command to install support for Networking:

```
make install-network
```

11.2.2. Contents of Bootscripts

Installed scripts: checkfs, cleanfs, console, functions, halt, ifdown, ifup, localnet, mountfs, mountkernfs, network, rc, reboot, sendsignals, setclock, static, swap, sysklogd, template, 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>/var/run/</code> and <code>/var/lock/</code> ; it re-creates <code>/var/run/utmp</code> and removes the possibly present <code>/etc/nologin</code> , <code>/fastboot</code> , and <code>/forcefsck</code> files
console	Loads the correct keymap table for the desired keyboard layout; it also sets the screen font
functions	Contains common functions, such as error and status checking, that are used by several bootscripts
halt	Halts the system
ifdown	Assists the network script with stopping network devices
ifup	Assists the network script with starting network devices
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>
network	Sets up network interfaces, such as network cards, and sets up the default gateway (where applicable)
rc	The master run-level control script; it is responsible for running all the other bootscripts one-by-one, in a sequence determined by the name of the symbolic links being processed
reboot	Reboots the system
sendsignals	Makes sure every process is terminated before the system reboots or halts
setclock	Resets the kernel clock to local time in case the hardware clock is not set to UTC time
static	Provides the functionality needed to assign a static Internet Protocol (IP) address to a network interface
swap	Enables and disables swap files and partitions

sysklogd	Starts and stops the system and kernel log daemons
template	A template to create custom bootscripts for other daemons
udev	Starts and stops the Eudev daemon

11.3. How Do These Bootscripts Work?

Linux uses a special booting facility named SysVinit that is based on a concept of *run-levels*. It can be quite different from one system to another, so it cannot be assumed that because things worked in one particular Linux distribution, they should work the same in CLFS too. CLFS has its own way of doing things, but it respects generally accepted standards.

SysVinit (which will be referred to as “init” from now on) works using a run-levels scheme. There are seven (numbered 0 to 6) run-levels (actually, there are more run-levels, but they are for special cases and are generally not used. See `init(8)` for more details), and each one of those corresponds to the actions the computer is supposed to perform when it starts up. The default run-level is 3. Here are the descriptions of the different run-levels as they are implemented:

```
0: halt the computer
1: single-user mode
2: multi-user mode without networking
3: multi-user mode with networking
4: reserved for customization, otherwise does the same as 3
5: same as 4, it is usually used for GUI login (like X's xdm or KDE's kdm)
6: reboot the computer
```

The command used to change run-levels is `init [runlevel]`, where `[runlevel]` is the target run-level. For example, to reboot the computer, a user could issue the `init 6` command, which is an alias for the `reboot` command. Likewise, `init 0` is an alias for the `halt` command.

There are a number of directories under `/etc/rc.d` that look like `rc?.d` (where `?` is the number of the run-level) and `rcsysinit.d`, all containing a number of symbolic links. Some begin with a *K*, the others begin with an *S*, and all of them have two numbers following the initial letter. The *K* means to stop (kill) a service and the *S* means to start a service. The numbers determine the order in which the scripts are run, from 00 to 99—the lower the number the earlier it gets executed. When `init` switches to another run-level, the appropriate services are either started or stopped, depending on the runlevel chosen.

The real scripts are in `/etc/rc.d/init.d`. They do the actual work, and the symlinks all point to them. Killing links and starting links point to the same script in `/etc/rc.d/init.d`. This is because the scripts can be called with different parameters like `start`, `stop`, `restart`, `reload`, and `status`. When a *K* link is encountered, the appropriate script is run with the `stop` argument. When an *S* link is encountered, the appropriate script is run with the `start` argument.

There is one exception to this explanation. Links that start with an *S* in the `rc0.d` and `rc6.d` directories will not cause anything to be started. They will be called with the parameter `stop` to stop something. The logic behind this is that when a user is going to reboot or halt the system, nothing needs to be started. The system only needs to be stopped.

These are descriptions of what the arguments make the scripts do:

```
start
```

The service is started.

```
stop
```

The service is stopped.

```
restart
```

The service is stopped and then started again.

`reload`

The configuration of the service is updated. This is used after the configuration file of a service was modified, when the service does not need to be restarted.

`status`

Tells if the service is running and with which PIDs.

Feel free to modify the way the boot process works (after all, it is your own CLFS system). The files given here are an example of how it can be done.

11.4. Configuring the `setclock` Script

The `setclock` script reads the time from the hardware clock, also known as the BIOS or the Complementary Metal Oxide Semiconductor (CMOS) clock. If the hardware clock is set to UTC, this script will convert the hardware clock's time to the local time using the `/etc/localtime` file (which tells the `hwclock` program which timezone the user is in). There is no way to detect whether or not the hardware clock is set to UTC, so this needs to be configured manually.

If you cannot remember whether or not the hardware clock is set to UTC, find out by running the `hwclock --localtime --show` command. This will display what the current time is according to the hardware clock. If this time matches whatever your watch says, then the hardware clock is set to local time. If the output from `hwclock` is not local time, chances are it is set to UTC time. Verify this by adding or subtracting the proper amount of hours for the timezone to the time shown by `hwclock`. For example, if you are currently in the MST timezone, which is also known as GMT -0700, add seven hours to the local time.

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

Create a new file `/etc/sysconfig/clock` by running the following:

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

UTC=1

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

A good hint explaining how to deal with time on CLFS is available at <http://hints.cross-lfs.org/index.php/time.txt>. It explains issues such as time zones, UTC, and the TZ environment variable.

11.5. Configuring the Linux Console

This section discusses how to configure the `i18n` bootscript that sets up the keyboard map and the console font. If non-ASCII characters (e.g., the British pound sign and Euro character) will not be used and the keyboard is a U.S. one, skip this section. Without the configuration file, the `console` bootscript will do nothing.

The `i18n` script reads the `/etc/sysconfig/i18n` file for configuration information. Decide which keymap and screen font will be used. Various language-specific HOWTO's can also help with this (see <http://www.tldp.org/HOWTO/HOWTO-INDEX/other-lang.html>). A pre-made `/etc/sysconfig/i18n` file with known settings for several countries was installed with the CLFS-Bootscripts package, so the relevant section can be uncommented if the country is supported. If still in doubt, look in the `/usr/share/consolefonts` for valid screen fonts and `/usr/share/keymaps` for valid keymaps.

The default `/etc/sysconfig/i18n` is set up for UTF-8 using the us keymap. You will need to edit the file to your specific needs. The `/etc/sysconfig/i18n` file has additional information in it to help you to assist in configuring.

11.6. Device and Module Handling on a CLFS System

In *Installing Basic System Software*, we installed the Eudev package. Before we go into the details regarding how this works, a brief history of previous methods of handling devices is in order.

Linux systems in general traditionally use a static device creation method, whereby a great many device nodes are created under `/dev` (sometimes literally thousands of nodes), regardless of whether the corresponding hardware devices actually exist. This is typically done via a **MAKEDEV** script, which contains a number of calls to the **mknod** program with the relevant major and minor device numbers for every possible device that might exist in the world.

Using the Eudev method, only those devices which are detected by the kernel get device nodes created for them. Because these device nodes will be created each time the system boots, they will be stored on a `tmpfs` file system (a virtual file system that resides entirely in system memory). Device nodes do not require much space, so the memory that is used is negligible.

11.6.1. History

In February 2000, a new filesystem called `devfs` was merged into the 2.3.46 kernel and was made available during the 2.4 series of stable kernels. Although it was present in the kernel source itself, this method of creating devices dynamically never received overwhelming support from the core kernel developers.

The main problem with the approach adopted by `devfs` was the way it handled device detection, creation, and naming. The latter issue, that of device node naming, was perhaps the most critical. It is generally accepted that if device names are allowed to be configurable, then the device naming policy should be up to a system administrator, not imposed on them by any particular developer(s). The `devfs` file system also suffers from race conditions that are inherent in its design and cannot be fixed without a substantial revision to the kernel. It has also been marked as deprecated due to a lack of recent maintenance.

With the development of the unstable 2.5 kernel tree, later released as the 2.6 series of stable kernels, a new virtual filesystem called `sysfs` came to be. The job of `sysfs` is to export a view of the system's hardware configuration to userspace processes. With this userspace-visible representation, the possibility of seeing a userspace replacement for `devfs` became much more realistic.

11.6.2. Eudev Implementation

11.6.2.1. Sysfs

The `sysfs` filesystem was mentioned briefly above. One may wonder how `sysfs` knows about the devices present on a system and what device numbers should be used for them. Drivers that have been compiled into the kernel directly register their objects with `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 and to **udev** for device node creation.

11.6.2.2. Eudev Bootscript

The **S10udev** initscript takes care of creating device nodes when Linux is booted. The script unsets the uevent handler from the default of `/sbin/hotplug`. This is done because the kernel no longer needs to call out to an external binary. Instead **udev** will listen on a netlink socket for uevents that the kernel raises. Next, the bootscript copies any static

device nodes that exist in `/lib/udev/devices` to `/dev`. This is necessary because some devices, directories, and symlinks are needed before the dynamic device handling processes are available during the early stages of booting a system. Creating static device nodes in `/lib/udev/devices` also provides an easy workaround for devices that are not supported by the dynamic device handling infrastructure. The bootscript then starts the Eudev daemon, **udev**, which will act on any uevents it receives. Finally, the bootscript forces the kernel to replay uevents for any devices that have already been registered and then waits for **udev** to handle them.

11.6.2.3. Device Node Creation

To obtain the right major and minor number for a device, Eudev relies on the information provided by `sysfs` in `/sys`. For example, `/sys/class/tty/vcs/dev` contains the string “7:0”. This string is used by **udev** to create a device node with major number 7 and minor 0. The names and permissions of the nodes created under the `/dev` directory are determined by rules specified in the files within the `/etc/udev/rules.d/` directory. These are numbered in a similar fashion to the CLFS-Bootscripts package. If **udev** can't find a rule for the device it is creating, it will default permissions to 660 and ownership to `root:root`. Documentation on the syntax of the Eudev rules configuration files is available in `/usr/share/doc/udev/writing_udev_rules/index.html`

11.6.2.4. Module Loading

Device drivers compiled as modules may have aliases built into them. Aliases are visible in the output of the **modinfo** program and are usually related to the bus-specific identifiers of devices supported by a module. For example, the `snd-fm801` driver supports PCI devices with vendor ID 0x1319 and device ID 0x0801, and has an alias of “`pci:v00001319d00000801sv*sd*bc04sc01i*`”. For most devices, the bus driver exports the alias of the driver that would handle the device via `sysfs`. E.g., the `/sys/bus/pci/devices/0000:00:0d.0/modalias` file might contain the string “`pci:v00001319d00000801sv00001319sd00001319bc04sc01i00`”. The default rules provided by Eudev 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.

11.6.2.5. Handling Hotpluggable/Dynamic Devices

When you plug in a device, such as a Universal Serial Bus (USB) MP3 player, the kernel recognizes that the device is now connected and generates a uevent. This uevent is then handled by **udev** as described above.

11.6.3. Problems with Loading Modules and Creating Devices

There are a few possible problems when it comes to automatically creating device nodes.

11.6.3.1. A kernel module is not loaded automatically

Eudev will only load a module if it has a bus-specific alias and the bus driver properly exports the necessary aliases to `sysfs`. In other cases, one should arrange module loading by other means. With Linux-3.10.14, Eudev 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 Eudev, 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 Eudev and expect the issue to be fixed later.

If there is no `modalias` file in the relevant directory under `/sys/bus`, this means that the kernel developers have not yet added `modalias` support to this bus type. With Linux-3.10.14, this is the case with ISA busses. Expect this issue to be fixed in later kernel versions.

Eudev is not intended to load “wrapper” drivers such as `snd-pcm-oss` and non-hardware drivers such as `loop` at all.

11.6.3.2. A kernel module is not loaded automatically, and Eudev 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 Eudev loads the wrapped module. To do this, add an “install” line in `/etc/modprobe.conf`. 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.

11.6.3.3. Eudev loads some unwanted module

Either don't build the module, or blacklist it in `/etc/modprobe.conf` file as done with the `forte` module in the example below:

```
blacklist forte
```

Blacklisted modules can still be loaded manually with the explicit **modprobe** command.

11.6.3.4. Eudev creates a device incorrectly, or 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**.

11.6.3.5. Eudev 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.

11.6.3.6. Eudev does not create a device

Further text assumes that the driver is built statically into the kernel or already loaded as a module, and that you have already checked that Eudev doesn't create a misnamed device.

Eudev has no information needed to create a device node if a kernel driver does not export its data to `sysfs`. This is most common with third party drivers from outside the kernel tree. Create a static device node in `/lib/udev/devices` with the appropriate major/minor numbers (see the file `devices.txt` inside the kernel documentation or the documentation provided by the third party driver vendor). The static device node will be copied to `/dev` by the **S10udev** bootscript.

11.6.3.7. Device naming order changes randomly after rebooting

This is due to the fact that Eudev, 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 Eudev. See Section 11.7, “Creating custom symlinks to devices” and Networking Configuration for examples.

11.6.4. Useful Reading

Additional helpful documentation is available at the following sites:

- A Userspace Implementation of devfs
http://www.kroah.com/linux/talks/ols_2003_udev_paper/Reprint-Kroah-Hartman-OLS2003.pdf
- The sysfs Filesystem
<http://www.kernel.org/pub/linux/kernel/people/mochel/doc/papers/ols-2005/mochel.pdf>

11.7. Creating custom symlinks to devices

11.7.1. 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 Eudev does not recognize the backslash for line continuation. If modifying Eudev 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/sdX`, 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.

11.7.2. Dealing with duplicate devices

As explained in Section 11.6, “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.

More information on writing Eudev rules can be found in `/usr/share/doc/udev/writing_udev_rules/index.html`.

11.8. 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. In the next section, a base `/etc/profile` will be created to set up locale information.

11.9. Setting Up Locale Information

The base `/etc/profile` below sets some environment variables necessary for native language support. Setting them properly results in:

- The output of programs translated into the native language
- Correct classification of characters into letters, digits and other classes. This is necessary for **bash** to properly accept non-ASCII characters in command lines in non-English locales
- The correct alphabetical sorting order for the country
- Appropriate default paper size
- Correct formatting of monetary, time, and date values

This script also sets the `INPUTRC` environment variable that makes Bash and Readline use the `/etc/inputrc` file created earlier.

Replace `[LL]` below with the two-letter code for the desired language (e.g., “en”) and `[CC]` with the two-letter code for the appropriate country (e.g., “GB”). `[charmap]` should be replaced with the canonical charmap for your chosen locale.

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 the Bash startup files:

```
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/profile` file:

```
cat > /etc/profile << "EOF"
# Begin /etc/profile

export LANG=[ll]_[CC].[charmap]
export INPUTRC=/etc/inputrc

# End /etc/profile
EOF
```

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.cross-lfs.org/>.

11.10. 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
```

Chapter 12. Networking Configuration

12.1. Configuring the localnet Script

Part of the job of the `localnet` script is setting the system's hostname. This needs to be configured in the `/etc/sysconfig/network` file.

Create the `/etc/sysconfig/network` file and enter a hostname by running:

```
echo "HOSTNAME=[clfs]" > /etc/sysconfig/network
```

`[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, 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:

Class	Networks
A	10.0.0.0
B	172.16.0.0 through 172.31.0.255
C	192.168.0.0 through 192.168.255.255

A valid IP address could be 192.168.1.1. A valid FQDN for this IP could be `www.linuxfromscratch.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, an FQDN is still required. This is necessary for certain programs to operate correctly.

Create the `/etc/hosts` file by running:

```
cat > /etc/hosts << "EOF"
# Begin /etc/hosts (network card version)

127.0.0.1 localhost
[192.168.1.1] [<HOSTNAME>.example.org] [HOSTNAME]

# 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).

If a network card is not going to be configured, create the `/etc/hosts` file by running:

```
cat > /etc/hosts << "EOF"
# Begin /etc/hosts (no network card version)

127.0.0.1 [<HOSTNAME>.example.org] [HOSTNAME] localhost

# End /etc/hosts (no network card version)
EOF
```

12.3. Creating the `/etc/resolv.conf` File

12.3.1. 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
```

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. DHCP or Static 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. Dynamic will allow you to take advantage of a DHCP server to get all your configuration information. Static you become responsible for setting up your options.

To configure a Static Interface, Follow Section 12.5, “Static Networking Configuration”.

To configure a DHCP Interface, Follow Section 12.6, “DHCPD-6.1.0”.

12.5. Static Networking Configuration

12.5.1. 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/network-devices` hierarchy. This directory should contain a sub-directory for each interface to be configured, such as `ifconfig.xyz`, where “xyz” is a network interface name. Inside this directory would be files defining the attributes to this interface, such as its IP address(es), subnet masks, and so forth.

The following command creates a sample `ipv4` file for the `eth0` device:

```
cd /etc/sysconfig/network-devices &&
mkdir -v ifconfig.eth0 &&
cat > ifconfig.eth0/ipv4 << "EOF"
ONBOOT="yes"
SERVICE="ipv4-static"
IP="192.168.1.1"
GATEWAY="192.168.1.2"
PREFIX="24"
BROADCAST="192.168.1.255"
EOF
```

The values of these variables must be changed in every file to match the proper setup. If the `ONBOOT` variable is set to “yes” the network script will bring up the Network Interface Card (NIC) during booting of the system. If set to anything but “yes” the NIC will be ignored by the network script and not be brought up.

The `SERVICE` variable defines the method used for obtaining the IP address. The CLFS-Bootscripts package has a modular IP assignment format, and creating additional files in the `/etc/sysconfig/network-devices/services` directory allows other IP assignment methods.

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.

To configure another DHCP Interface, Follow Section 12.7, “DHCP Networking Configuration”.

12.6. DHCPD-6.1.0

The DHCPD package provides a DHCP Client for network configuration.

12.6.1. 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
```

12.6.2. Contents of dhcpd

Installed files: dhcpd

Short Descriptions

dhcpd dhcpd 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.

12.7. DHCP Networking Configuration

12.7.1. Creating the DHCP Network Interface Configuration Files

First install the service from the CLFS Bootscripts package:

```
tar -xvf bootscripts-cross-lfs-2.1-pre1.tar.xz
cd bootscripts-cross-lfs-2.1-pre1
make install-service-dhcpd
```

Finally, create the `/etc/sysconfig/network-devices/ifconfig.eth0/dhcpd` configuration file using the following commands. Adjust appropriately for additional interfaces:

```
cd /etc/sysconfig/network-devices &&
mkdir -v ifconfig.eth0 &&
cat > ifconfig.eth0/dhcpd << "EOF"
ONBOOT="yes"
SERVICE="dhcpd"

# Start Command for DHCPD
DHCP_START="-q"

# Stop Command for DHCPD
DHCP_STOP="-k"
EOF
```

The values of these variables must be changed in every file to match the proper setup. If the `ONBOOT` variable is set to “yes” the network script will bring up the Network Interface Card (NIC) during booting of the system. If set to anything but “yes” the NIC will be ignored by the network script and not be brought up.

The `SERVICE` variable defines the method used for obtaining the IP address. The CLFS-Bootscripts package has a modular IP assignment format, and creating additional files in the `/etc/sysconfig/network-devices/services` directory allows other IP assignment methods.

The `DHCP_START` and `DHCP_STOP` variables arguments that are passed onto `dhcpd` when starting and stopping the service. More information about what can be passed can be found in the `dhcpd(8)` man page.

To configure another Static Interface, Follow Section 12.5, “Static Networking Configuration”.

Chapter 13. Making the CLFS System Bootable

13.1. Introduction

It is time to make the CLFS system bootable. This chapter discusses creating an `fstab` file, 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. Creating the `/etc/fstab` File

The `/etc/fstab` file is used by some programs to determine where file systems are to be mounted by default, in which order, and which must be checked (for integrity errors) prior to mounting. Create a new file systems table like this:

```
cat > /etc/fstab << "EOF"
# Begin /etc/fstab

# file system  mount-point  type  options                dump  fsck
#                                     order

/dev/[xxx]    /             [fff] defaults                1     1
/dev/[yyy]    swap          swap  pri=1                   0     0
proc          /proc         proc  defaults                0     0
sysfs         /sys          sysfs defaults                0     0
devpts        /dev/pts      devpts gid=5,mode=620          0     0
shm           /dev/shm      tmpfs defaults                0     0
tmpfs         /run          tmpfs defaults                0     0
devtmpfs      /dev          devtmpfs mode=0755,nosuid 0     0

# End /etc/fstab
EOF
```

Replace `[xxx]`, `[yyy]`, and `[fff]` with the values appropriate for the system, for example, `hda2`, `hda5`, and `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.

13.3. Linux-3.10.14

The Linux package contains the Linux kernel.

13.3.1. Installation of the kernel

Building the kernel involves a few steps—configuration, compilation, and installation. Read the `README` file in the kernel source tree for alternative methods to the way this book configures the kernel.

Prepare for compilation by running the following command:

```
make mrproper
```

This ensures that the kernel tree is absolutely clean. The kernel team recommends that this command be issued prior to each kernel compilation. Do not rely on the source tree being clean after un-tarring.

Configure the kernel via a menu-driven interface. Please note that the `udev` bootscript requires `"rtc"`, `"tmpfs"` and `"devtmpfs"` to be enabled and built into the kernel, not as modules. CBLFS has some information regarding particular kernel configuration requirements of packages outside of CLFS at <http://cblfs.cross-lfs.org/>:

```
make menuconfig
```

Alternatively, `make oldconfig` may be more appropriate in some situations. See the `README` file for more information.



Warning

If you are using an existing config in which the `ARCH` was specified as `ppc` (instead of `powerpc`), you will have to run `make menuconfig` after `make oldconfig` and manually select many of the mac-specific options for `ide` and `input`.

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, an `/etc/modprobe.conf` 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.conf(5)` may be of interest.

Be very careful when reading other documentation relating to kernel modules because it usually applies to 2.4.x kernels only. As far as we know, kernel configuration issues specific to `Hotplug` and `Eudev` are not documented. The problem is that `Eudev` will create a device node only if `Hotplug` or a user-written script inserts the corresponding module into the kernel, and not all modules are detectable by `Hotplug`. Note that statements like the one below in the `/etc/modprobe.conf` file do not work with `Eudev`:

```
alias char-major-XXX some-module
```

Because of the complications with `Eudev` and modules, we strongly recommend starting with a completely non-modular kernel configuration, especially if this is the first time using `Eudev`.

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/clfskernel-3.10.14
```

`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-3.10.14
```

The kernel configuration file `.config` produced by the **make menuconfig** step above contains all the configuration selections for the kernel that was just compiled. It is a good idea to keep this file for future reference:

```
cp -v .config /boot/config-3.10.14
```

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-3.10.14` 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.

13.3.2. Contents of Linux

Installed files: `config-[linux-version]`, `clfskernel-[linux-version]`, and `System.map-[linux-version]`
Installed directory: `/lib/modules`

Short Descriptions

`config-[linux-version]` Contains all the configuration selections for the kernel

- `clfskernel-[linux-version]` The engine of the Linux system. When turning on the computer, the kernel is the first part of the operating system that gets loaded. It detects and initializes all components of the computer's hardware, then makes these components available as a tree of files to the software and turns a single CPU into a multitasking machine capable of running scores of programs seemingly at the same time.
- `System.map-[linux-version]` A list of addresses and symbols; it maps the entry points and addresses of all the functions and data structures in the kernel

13.4. Making the CLFS System Bootable

Your shiny new CLFS system is almost complete. One of the last things to do is to ensure that the system can be properly booted. The instructions below apply only to NewWorld Macintoshes.

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 CD is ready to “rescue” the computer if it becomes un-bootable. It is also a good idea to enable booting from Open Firmware in case things go really wrong.

Earlier, we compiled and installed the yaboot boot loader software in preparation for this step. The procedure involves writing the bootloader to a bootstrap partition and blessing it so that Open Firmware will boot from it. This is all handled by **ybin**, the yaboot installer.

Ybin writes an optional 'OS selector' menu into Open Firmware, then writes yaboot and yaboot.conf to the bootstrap partition, blesses this, and updates the boot device recorded in nvram. When booted, the OF provides the initial menu to choose between linux, boot from CD, and e.g. OSX (depending on what was in yaboot.conf). If you boot to 'linux', yaboot is executed and lets you select which kernel to use.

Images (kernels) are specified, together with any necessary path, in yaboot.conf - the details are incorporated into the bootloader, but no attempt is made to access or validate the paths until they are selected. There are many possible options that can be specified in yaboot.conf, see the man page for the details. Most people will be able to specify device=hd: (for a single hard disk), but if you have multiple disks, or if you wish to be pedantic, you can specify the full OF path to the device, obtained by running **ofpath /dev/hdX**.

Using the above information, determine the appropriate designators for the bootstrap partition and the root partition. For the following example, it is assumed that the bootstrap partition is hda2 and the root partition is hda7. We will also assume that you wish to be able to boot an OSX installation on hda4. Change these items as necessary for your machine.

If your machine has a SATA disk, specify the partitions using /dev/sda7 and so forth in the usual way. At least some of the distros specify a full OF path to the 'device' and to the image(s), such as *device=/ht@0,f2000000/pci@3/k2-sata-root@c/k2-sata@0/disk@0:* for the disk, and *image=/ht@0,f2000000/pci@3/k2-sata-root@c/k2-sata@0/disk@0:9,/boot/clfskernel-3.10.14* which definitely works.

Create a “yaboot.conf” file defining yaboot's boot menu:

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

# By default, yaboot will boot the first menu entry.

# Allow 10 seconds before booting the default.
# this will also apply to the first-stage os selector
timeout=100

# These variables are global
# first, where to put the bootstrap partition
boot=/dev/hda2
```

```

# Which disk to use
device=hd:

# Default partition for the kernel images
partition=7

# default root partition
root=/dev/hda7

# where ybin is to find yaboot and ofboot
install=/usr/lib/yaboot/yaboot
magicboot=/usr/lib/yaboot/ofboot

# allow the initial menu to offer CD as an option
enablecdboot

# allow the initial menu to offer booting from Open Firmware
enableofboot

# allow the initial menu to boot from mac osx
macosx=/dev/hda4

# white on black is boring!
# note the spellings : 'fgcolor' but 'light'
# in this context, light means 'without high intensity'
fgcolor=light-green

# The first entry is for CLFS.
# For all images, the pathname is relative to the filesystem
# on which they are situated and can include at most one
# directory
image=/boot/clfskernel-3.10.14
    label=2.1.0
    read-only
EOF

```

Add an entry for the host distribution, if you have one. It might look something like this if the kernel and initrd are in the host's '/' directory on hda6:

```

cat >> /etc/yaboot.conf << "EOF"
title Debian
image=/pci@f4000000/ata-6d/disk@0:6,/vmlinux
    label=Debian
    initrd=/pci@f4000000/ata-6d/disk@0:6,/initrd.gz
    initrd-size=10000
    append="root=/dev/hda7"
    read-only
EOF

```

**Warning**

The following command will update the bootstrap partition and the boot variable in Open Firmware. Do not run the command if this is not desired.

```
ybin
```

Alternatively, if the bootstrap partition has not already been initialized, perhaps because you are using a Live CD, you will need to use a different command to install the bootloader for the first time:

```
mkofboot
```

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 2.1.0 > /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.cross-lfs.org/index.php/Curl>
- Inetutils <http://cblfs.cross-lfs.org/index.php/Inetutils>
- LFTP <http://lftp.yar.ru/>
- Links <http://cblfs.cross-lfs.org/index.php/Links>
- Lynx <http://cblfs.cross-lfs.org/index.php/Lynx>
- NcFTP Client <http://cblfs.cross-lfs.org/index.php/Ncftp>
- Wget <http://cblfs.cross-lfs.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');
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/pub/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, lets 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
```

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 2.1.0* 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:

- Freecode (<http://freecode.com/>)

Freecode can notify you (via email) of new versions of packages installed on your system.

- *CERT* (Computer Emergency Response Team)

CERT has a mailing list that publishes security alerts concerning various operating systems and applications. Subscription information is available at <http://www.us-cert.gov/cas/signup.html>.

- Bugtraq

Bugtraq is a full-disclosure computer security mailing list. It publishes newly discovered security issues, and occasionally potential fixes for them. Subscription information is available at <http://www.securityfocus.com/archive>.

- 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.cross-lfs.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.cross-lfs.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 (TLDP) is to collaborate on all of the issues of Linux documentation. The TLDP features a large collection of HOWTOs, guides, and man pages. It is located at <http://www.tldp.org/>.

Part VI. Appendices

Appendix A. Acronyms and Terms

ABI	Application Binary Interface
ALSA	Advanced Linux Sound Architecture
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 testsuites. 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.

Autoconf

Installation depends on: Bash, Coreutils, Gawk, Grep, M4, Make, Perl, Sed and Texinfo
Test suite depends on: Automake, Binutils, Diffutils, Findutils, GCC and Libtool
Must be installed before: Automake

Automake

Installation depends on: Autoconf, Bash, Binutils, Coreutils, Gawk, Grep, M4, Make, Perl, Sed and Texinfo
Test suite depends on: Bison, Bzip2, DejaGNU, Diffutils, Expect, Findutils, Flex, GCC, Gettext, Gzip, Libtool, XZ-Utils and Tar. 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, EGLIBC, Gawk, GCC, Grep, Make, Ncurses, Patch, Readline, Sed and Texinfo
Test suite depends on: None
Must be installed before: None

Bc

Installation depends on: Bash, Binutils, Bison, Coreutils, EGLIBC, GCC, Grep, Make, and Readline
Test suite depends on: Gawk
Must be installed before: None

Binutils

Installation depends on: Bash, Binutils, Coreutils, Diffutils, EGLIBC, File, Gawk, GCC, Grep, Make, Perl, Sed, Texinfo and Zlib
Test suite depends on: DejaGNU and Expect
Must be installed before: None

Bison

Installation depends on: Bash, Binutils, Coreutils, EGLIBC, Gawk, GCC, Grep, M4, Make and Sed
Test suite depends on: Diffutils, Findutils and Gawk
Must be installed before: Flex, Kbd and Tar

Bzip2

Installation depends on: Bash, Binutils, Coreutils, EGLIBC, GCC, Make
Test suite depends on: Diffutils
Must be installed before: None

CLFS-Bootscripts

Installation depends on: Bash, Coreutils, Make and Sed
Test suite depends on: None
Must be installed before: None

Check

Installation depends on: GCC, Grep, Make, Sed and Texinfo
Test suite depends on: None
Must be installed before: None

CLooG-ISL

Installation depends on: Bash, Binutils, Coreutils, Diffutils, EGLIBC, Gawk, GCC, Grep, GMP, Make, MPC, MPFR, Sed and Texinfo
Test suite depends on: None
Must be installed before: GCC

Coreutils

Installation depends on: Bash, Binutils, Coreutils, EGLIBC, Gawk, GCC, GMP, Grep, Make, Patch, Perl, Sed and Texinfo
Test suite depends on: Diffutils, E2fsprogs, Findutils, Util-linux
Must be installed before: Bash, Diffutils, Findutils, Man and Eudev

DejaGNU

Installation depends on: Bash, Coreutils, Diffutils, GCC, Grep, Make and 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 testsuite available
Must be installed before: None

Diffutils

Installation depends on: Bash, Binutils, Coreutils, EGLIBC, GCC, Grep, Make, Patch, Sed and Texinfo
Test suite depends on: No testsuite available
Must be installed before: None

EGLIBC

Installation depends on: Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Gettext, Grep, Gzip, Make, Perl, Sed and Texinfo
Test suite depends on: None
Must be installed before: None

Expect

Installation depends on: Bash, Binutils, Coreutils, Diffutils, EGLIBC, GCC, Grep, Make, Patch, Sed and Tcl
Test suite depends on: None
Must be installed before: None

E2fsprogs

Installation depends on: Bash, Binutils, Coreutils, EGLIBC, Gawk, GCC, Gettext, Grep, Gzip, Make, Pkg-config-lite, Sed, Texinfo and Util-linux
Test suite depends on: Bzip2 and Diffutils
Must be installed before: None

File

Installation depends on: Bash, Binutils, Coreutils, Diffutils, EGLIBC, Gawk, GCC, Grep, Make, Sed and Zlib
Test suite depends on: No testsuite available
Must be installed before: None

Findutils

Installation depends on: Bash, Binutils, Coreutils, EGLIBC, GCC, Grep, Make, Sed and Texinfo
Test suite depends on: DejaGNU, Diffutils and Expect
Must be installed before: None

Flex

Installation depends on: Bash, Binutils, Coreutils, EGLIBC, GCC, Grep, M4, Make, Sed and Texinfo
Test suite depends on: Bison, Diffutils and Gawk
Must be installed before: IPRoute2, Kbd and Man

Gawk

Installation depends on: Bash, Binutils, Coreutils, EGLIBC, GCC, Grep, Make, Sed and Texinfo
Test suite depends on: Diffutils
Must be installed before: None

Gcc

Installation depends on: Bash, Binutils, CLoG-ISL, Coreutils, Diffutils, EGLIBC, Findutils, Gawk, GCC, GMP, Grep, ISL, Make, MPFR, Patch, Perl, Sed, Tar and Texinfo

Test suite depends on: Check, DejaGNU, and Expect

Must be installed before: None

Gettext

Installation depends on: Bash, Binutils, Coreutils, Diffutils, EGLIBC, Findutils, Gawk, GCC, Grep, Make, Sed and Texinfo

Test suite depends on: Tar and Tcl

Must be installed before: Automake

Glib

Installation depends on: bash, binutils, coreutils, gawk, gcc, gettext, make & M4.

Test suite depends on: Unknown

Must be installed before: Pkg-config-lite

GMP

Installation depends on: Bash, Binutils, Coreutils, Diffutils, EGLIBC, Gawk, GCC, Grep, M4, Make, Sed and Texinfo

Test suite depends on: None

Must be installed before: MPFR, GCC

Grep

Installation depends on: Bash, Binutils, Coreutils, EGLIBC, GCC, Grep, Make, Patch, Sed and Texinfo

Test suite depends on: Diffutils and Gawk

Must be installed before: Man

Groff

Installation depends on: Bash, Binutils, Coreutils, EGLIBC, Gawk, GCC, Grep, Make, Perl Sed and Texinfo

Test suite depends on: No testsuite available

Must be installed before: Man and Perl

Gzip

Installation depends on: Bash, Binutils, Coreutils, EGLIBC, GCC, Grep, Make, Sed and Texinfo

Test suite depends on: Diffutils

Must be installed before: Man

iana-Etc

Installation depends on: Coreutils, Gawk and Make

Test suite depends on: No testsuite available

Must be installed before: Perl

IProute2

- Installation depends on:** Bash, Binutils, Bison, Coreutils, EGLIBC, Findutils, Flex, GCC, Make, Linux-Headers and Sed
- Test suite depends on:** No testsuite available
- Must be installed before:** None

IPutils

- Installation depends on:** Bash, Binutils, Coreutils, EGLIBC, GCC and Make
- Test suite depends on:** No testsuite available
- Must be installed before:** None

ISL

- Installation depends on:** Bash, Binutils, Coreutils, Diffutils, EGLIBC, Gawk, GCC, Grep, GMP, Make, MPC, MPFR, Sed and Texinfo
- Test suite depends on:** None
- Must be installed before:** GCC

Kbd

- Installation depends on:** Bash, Binutils, Coreutils, EGLIBC, Gawk, GCC, Gzip, Make, and Check
- Test suite depends on:** No testsuite available
- Must be installed before:** None

KMOD

- Installation depends on:** Bash, Binutils, Bison, Coreutils, EGLIBC, Flex, Gawk, GCC, Gettext, Gzip, Make, Pkg-config-lite, Sed, XZ-Utils, and Zlib.
- Test suite depends on:** No testsuite available
- Must be installed before:** Eudev

Less

- Installation depends on:** Bash, Binutils, Coreutils, EGLIBC, GCC, Grep, Make, Ncurses and Sed
- Test suite depends on:** No testsuite available
- Must be installed before:** None

Libee

- Installation depends on:** Bash, Binutils, Coreutils, Diffutils, EGLIBC, Findutils, Gawk, GCC, Grep, Libestr, Make, Pkg-config-lite, Sed and Texinfo
- Test suite depends on:** None
- Must be installed before:** Rsyslog

Libestr

- Installation depends on:** Bash, Binutils, Coreutils, Diffutils, EGLIBC, Findutils, Gawk, GCC, Grep, Make, Sed and Texinfo
- Test suite depends on:** None
- Must be installed before:** Libee and Rsyslog

Libtool

- Installation depends on:** Bash, Binutils, Coreutils, Diffutils, EGLIBC, Findutils, Gawk, GCC, Grep, Make, Sed and Texinfo
- Test suite depends on:** Autoconf
- Must be installed before:** None

Linux-Headers

- Installation depends on:** Binutils, Coreutils, Findutils, GCC, Grep, Make, Perl and Sed
- Test suite depends on:** No testsuite available
- Must be installed before:** None

Linux Kernel

- Installation depends on:** Bash, Binutils, Coreutils, Diffutils, EGLIBC, Findutils, GCC, Grep, Gzip, Make, KMOD, Ncurses, Perl and Sed
- Test suite depends on:** No testsuite available
- Must be installed before:** None

M4

- Installation depends on:** Bash, Binutils, Coreutils, EGLIBC, Gawk, GCC, Grep, Make, Sed and Texinfo
- Test suite depends on:** Diffutils
- Must be installed before:** Autoconf and Bison

Make

- Installation depends on:** Bash, Binutils, Coreutils, EGLIBC, GCC, Grep, Make, Sed and Texinfo
- Test suite depends on:** Perl and Procps
- Must be installed before:** None

Man

- Installation depends on:** Bash, Binutils, Bzip2, Coreutils, EGLIBC, Gawk, GCC, Grep, Groff, Gzip, Less, XZ-Uutils, Make and Sed
- Test suite depends on:** No testsuite available
- Must be installed before:** None

Man-Pages

- Installation depends on:** Bash, Coreutils, and Make
- Test suite depends on:** No testsuite available
- Must be installed before:** None

MPC

- Installation depends on:** Bash, Binutils, Coreutils, Diffutils, EGLIBC, Gawk, GCC, Grep, GMP, Make, MPFR, Sed and Texinfo
- Test suite depends on:** None
- Must be installed before:** GCC

MPFR

- Installation depends on:** Bash, Binutils, Coreutils, Diffutils, EGLIBC, Gawk, GCC, Grep, GMP, Make, Sed and Texinfo
- Test suite depends on:** None
- Must be installed before:** GCC

KMOD

- Installation depends on:** Bash, Binutils, Coreutils, EGLIBC, Findutils, GCC, Grep, Make, Sed and Zlib
- Test suite depends on:** Diffutils, File, Gawk and Gzip
- Must be installed before:** None

Ncurses

- Installation depends on:** Bash, Binutils, Coreutils, Diffutils, EGLIBC, Gawk, GCC, Grep, Make and Sed
- Test suite depends on:** No testsuite available
- Must be installed before:** Bash, GRUB, Inetutils, Less, Procps, Psmisc, Readline, Texinfo, Util-linux and Vim

Patch

- Installation depends on:** Bash, Binutils, Coreutils, EGLIBC, GCC, Grep, Make and Sed
- Test suite depends on:** No testsuite available
- Must be installed before:** None

Perl

- Installation depends on:** Bash, Binutils, Coreutils, EGLIBC, Gawk, GCC, Grep, Make and Sed
- Test suite depends on:** Gzip, Iana-Etc and Procps, Tar
- Must be installed before:** Autoconf

Pkg-config-lite

- Installation depends on:** Bash, Binutils, Coreutils, Diffutils, EGLIBC, Gawk, GCC, Grep, Make and Sed
- Test suite depends on:** None
- Must be installed before:** Util-linux, E2fsprogs

Procps

- Installation depends on:** Bash, Binutils, Coreutils, EGLIBC, GCC, Make and Ncurses
- Test suite depends on:** No testsuite available
- Must be installed before:** None

Psmisc

- Installation depends on:** Bash, Binutils, Coreutils, EGLIBC, GCC, Grep, Make, Ncurses and Sed
- Test suite depends on:** No testsuite available
- Must be installed before:** None

Readline

- Installation depends on:** Bash, Binutils, Coreutils, EGLIBC, GCC, Grep, Make, Ncurses, Patch, Sed and Texinfo
- Test suite depends on:** No testsuite available
- Must be installed before:** Bash

Rsyslog

- Installation depends on:** Binutils, Coreutils, Diffutils, EGLIBC, Gawk, GCC, Grep, libee, Libestr, Make, Sed and Zlib
- Test suite depends on:** No testsuite available
- Must be installed before:** None

Sed

- Installation depends on:** Bash, Binutils, Coreutils, EGLIBC, GCC, Grep, Make, Sed and Texinfo
- Test suite depends on:** Diffutils and Gawk
- Must be installed before:** E2fsprogs, File, Libtool and Shadow

Shadow

- Installation depends on:** Bash, Binutils, Coreutils, Diffutils, EGLIBC, Findutils, Gawk, GCC, Gettext, Grep, Make and Sed
- Test suite depends on:** No testsuite available
- Must be installed before:** None

Sysvinit

- Installation depends on:** Binutils, Coreutils, EGLIBC, GCC, Make and Sed
- Test suite depends on:** No testsuite available
- Must be installed before:** None

Tar

- Installation depends on:** Bash, Binutils, Bison, Coreutils, EGLIBC, GCC, Grep, Make, Sed and Texinfo
- Test suite depends on:** Diffutils, Findutils, Gawk and Gzip
- Must be installed before:** None

Tcl

- Installation depends on:** Bash, Binutils, Coreutils, Diffutils, EGLIBC, GCC, Grep, Make and Sed
- Test suite depends on:** None
- Must be installed before:** None

Texinfo

- Installation depends on:** Bash, Binutils, Coreutils, EGLIBC, Gawk, GCC, Grep, Make, Ncurses and Sed
- Test suite depends on:** Diffutils and Gzip
- Must be installed before:** None

Eudev

Installation depends on: Binutils, Coreutils, Diffutils, EGLIBC, Gawk, GCC, Grep, Make and Sed
Test suite depends on: No testsuite available
Must be installed before: None

Util-linux

Installation depends on: Bash, Binutils, Coreutils, EGLIBC, GCC, Grep, Make, Ncurses, Pkg-config-lite, Sed, Texinfo and Zlib
Test suite depends on: No testsuite available
Must be installed before: E2fsprogs

Vim

Installation depends on: Bash, Binutils, Coreutils, Diffutils, EGLIBC, Findutils, Gawk, GCC, Gettext, Grep, Make, Ncurses, Perl and Sed
Test suite depends on: Gzip
Must be installed before: None

XZ-Utills

Installation depends on: Bash, Binutils, Coreutils, Diffutils, EGLIBC, Findutils, Gawk, GCC, Grep, Make and Sed
Test suite depends on: None
Must be installed before: None

Zlib

Installation depends on: Bash, Binutils, Coreutils, EGLIBC, GCC, Make and Sed
Test suite depends on: None
Must be installed before: File, KMOD and Util-linux

Appendix C. PowerPC Dependencies

This page contains dependency information for packages specific to ppc.

hfsutils

Installation depends on: Bash, Binutils, Coreutils, GCC and Make

Test suite depends on: None

Must be installed before: None

Parted

Installation depends on: Bash, Binutils, Coreutils, E2fsprogs, GCC, Make, Ncurses and Readline

Test suite depends on: None

Must be installed before: None

Powerpc-Utills

Installation depends on: Binutils, GCC, Make and Patch

Test suite depends on: None

Must be installed before: None

Yaboot

Installation depends on: Binutils, Coreutils, GCC, Make, Mktmp, Patch and Sed

Test suite depends on: None

Must be installed before: None

Appendix D. Open Firmware and Mac issues.

This appendix documents some of the features of ppc macintoshes, and in particular the requirements of coexisting with Mac OS's (OSX or the old OS9). It is only relevant to NewWorld hardware.

Open Firmware and blessed partitions

The Open Firmware (OF) is the code in ROM or nvram which controls how the machine boots. If booting automatically, it will boot from the first valid blessed partition it finds (this is a simplification, but it is adequate for normal purposes).

It can only read apple filesystems (hfs, hfs+, or hfsx depending on the version of the firmware). For disks under linux, the blessing is done by ybin when it installs yaboot (the loader) and yaboot.conf.

Mac OS's have a tendency to look at other hfs{+,x} filesystems on the disk, and unbless them if they do not match their expectations. Unblessing makes them unbootable. Fortunately, a filesystem of type `Apple_Bootstrap` can be read as hfs by the OF, but will be ignored by Mac OS.

Partitioning

Macintoshes use their own partition format - this means that other machines are unlikely to be able to read or write to macintosh partitions (in particular, fdisk does not understand them). The format allows a large number of individual partitions, and the native Mac tools had a tendency to insert small "filler" partitions between the real partitions. Under linux, using more than 15 partitions can be problematic (shortage of device nodes), so the normal approach is to use the Mac tools to create an area of freespace at the *front* of the disk, then put the Mac OS partition(s) after it and (re-)install the Mac OS. The freespace can then be partitioned using **parted** or the older **mac-fdisk**. It seems that recent versions of the Mac tools may no longer insert the filler partitions, so it may be possible to do all the partitioning before installing OSX.



Warning

The Macintosh resizing and partitioning tools are destructive and may delete all data when a partition is resized, even on unaltered partitions.

For the Linux partitions, you will need a bootstrap partition - this can normally be a mere 800KB in size (the smallest hfs partition available) although the Fedora installer has been known to insist on 800MB. This has to be in front of the Mac OS partition. The bootstrap is *never* mounted as a regular partition and should not be confused with a `/boot` partition. Other partitions are as normal (at least one rootfs, perhaps swap, perhaps others).

According to the `lfs-from-osx` hint, the Mac partitioning tools can create an `apple_bootstrap` partition and therefore there is no need to use a Linux CD to create the desired partitions from freespace, but using a Linux CD to create the partitions is a more widely tested approach.

If you follow this approach, partition 1 will be the apple partition map, partition 2 will be the bootstrap at the start of the disk, the linux partitions will follow, and then the mac partition(s) - under OSX the first mac partition will be number 3, under OS9 it would have a higher number and there would be some apple driver partitions.

OSX or OF upgrades

If the machine is dual-booted with OSX, the mac kernel or the OF will probably be upgraded at some point. This appears to either unbless the bootstrap, or else just point the OF boot device to the mac partition - so, the linux system will no longer be bootable.

Therefore, you will need to know which partition contains the bootstrap so that you can boot it from OF (on an apple keyboard, hold down option-command-o-f (that is, alt-apple-o-f) while booting then enter a command like:

```
boot hd:2,yaboot
```

This will allow you to select a linux boot, and from there you will have to rerun **ybin**.

The "OS chooser" menu that yaboot typically loads is stored in the OF and will not be available after a Mac kernel or firmware upgrade until **ybin** has been rerun.

Yaboot's requirements

Yaboot is the boot loader for linux, sometimes referred to as the second stage loader. It reads the yaboot.conf file on the bootstrap partition to find which linux system(s) should be available, and attempts to load the required kernel.

The bootstrap man page warns that the path to the kernel should contain no more than one directory for reliability. Yaboot has to be able to understand the filesystem, so that it can find the kernel. It understands hfs (not useful for linux, it is not case-sensitive), ext2 (and therefore it can read ext3), reiser3, and xfs. If you want to use a different type of filesystem for '/' you will have to create a separate boot partition with a supported filesystem, and use that to hold the kernels.

Requirements if starting from OSX

Older versions of OSX (panther, leopard) can write to ext2 filesystems using version 1.3 of ext2fsx. The upgrade to tiger broke this, and version 1.4 of ext2fsx only supports reading. Users of current OSX will therefore have to find some other way of creating a suitable filesystem and populating it, such as a Live CD or rescue CD.

Appendix E. 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.

- 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-Bootscripts

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.

- CLooG-ISL

This package is used by GCC.

- Coreutils

This package contains many basic command-line file-management tools, required for installation of every package in CLFS.

- DejaGNU

This package is needed for the testsuites 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.

- EGLIBC

Any dynamically-linked C program (which is nearly everything in CLFS) needs a C library to compile and run.

- Expect

This package is needed for the testsuites 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. 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.

- Gettext

A tool that allows programmers to easily implement i18n (internationalization) in their programs. It is a required dependency for a number of packages

- GMP

This package is required by GCC.

- 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.

- 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.

- IProute2

This package contains programs for administering network interfaces.

- IPutils

This package contains several basic network-management tools.

- ISL

This package is required by CLOoG.

- Kbd

Contains keytable files and keyboard utilities compatible with the Linux kernel.

- 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. Used by Man for displaying manpages.

- Libee

This package contains an event expression library. It is needed by Rsyslog.

- Libestr

This package contains a library for string essentials. It is needed by Rsyslog.

- Libtool

The Libtool package contains the GNU generic library support script. It is used by some CLFS packages.

- Linux-Headers

This package consists of santized 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

Required for installation of most CLFS packages

- Man

Used for viewing manpages

- 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
Needed by E2fsprogs
- Procps
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.
- Rsyslog
Rsyslog is an enhanced multi-threaded syslogd that supports multiple backends with very little dependencies. It provides a program that logs various system events into files in `/var/log`.
- 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.
- Sysvinit
Sysvinit is the init daemon that the clfs-bootscrips were written to work with.
- Tar
Required to unpack the tar archives in which all CLFS packages are distributed
- Tcl
Needed for the testsuites 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.

- Eudev

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

- 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.

- 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 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

- Autoconf: 183
- Automake: 184
- Bash: 186
 - temporary system: 64
- Bc: 187
 - cross-tools: 33
- Binutils: 145
 - cross tools: 43
 - temporary system: 60
- Bison: 174
 - temporary system: 66
- Bootscripts: 237
 - boot: 106
 - usage: 239
- Bzip2: 188
 - temporary system: 67
- Check: 123
- CLooG: 143
 - cross-tools: 42
 - temporary system: 58
- Coreutils: 167
 - temporary system: 68
- DejaGNU: 122
- DHCPCD: 253
- Diffutils: 190
 - temporary system: 69
- E2fsprogs: 161
 - boot: 91
- EGLIBC: 131
 - cross tools: 48
- Eudev: 226
 - boot: 95
 - usage: 241
- Expect: 121
- File: 191
 - cross-tools: 35
 - temporary system: 71
- Findutils: 193
 - temporary system: 70
- Flex: 176
 - temporary system: 72
- Gawk: 192
 - temporary system: 73
- GCC: 148
 - cross tools, final: 50
 - cross tools, static: 45
 - temporary system: 61
- Gettext: 195
 - temporary system: 74
- GMP: 139
 - cross-tools: 38
 - temporary system: 54
- Grep: 197
 - temporary system: 75
- Groff: 198
- Gzip: 202
 - temporary system: 76
- Hfsutils: 231
 - boot: 101
- Iana-Etc: 172
- IPRoute2: 177
- IPutils: 203
- ISL: 142
 - cross-tools: 41
 - temporary system: 57
- Kbd: 204
- Kmod: 211
 - boot: 94
- Less: 201
- Libee: 216
- Libestr: 215
- Libtool: 175
- Linux: 256
 - boot: 99
- Linux-Headers: 129
 - cross tools: 34
- M4: 173
 - temporary system: 36, 77
- Make: 206
 - temporary system: 78
- Man: 209
- Man-pages: 130
- MPC: 141
 - cross-tools: 40
 - temporary system: 56
- MPFR: 140
 - cross-tools: 39
 - temporary system: 55
- Ncurses: 152
 - cross-tools: 37

temporary system: 63
 Parted: 232
 Patch: 213
 temporary system: 79
 Perl: 179
 temporary tools: 128
 Pkg-config-lite: 154
 Powerpc-Utills: 233
 boot: 102
 Procps: 159
 Psmisc: 214
 Readline: 182
 rsyslog: 217
 configuring: 218
 Sed: 151
 temporary system: 80
 Shadow: 164
 boot: 90
 configuring: 165
 Sysvinit: 220
 boot: 92
 boot, configuring: 92
 configuring: 220
 Tar: 223
 temporary system: 81
 Tcl: 120
 Texinfo: 224
 temporary tools: 82
 Util-linux: 155
 boot: 89
 chroot: 110
 Vim: 228
 temporary system: 83
 XZ-Utills: 207
 temporary system: 85
 Yaboot: 234
 boot: 103
 boot, configuring: 107
 configuring: 259
 Zlib: 144
 boot: 59

Programs

a2p: 179, 180
 acinstall: 184, 184
 aclocal: 184, 184
 aclocal-1.12: 184, 184

addftinfo: 198, 198
 addnote: 234, 234
 addpart: 155, 156
 addr2line: 145, 146
 afmtodit: 198, 198
 agetty: 155, 156
 apropos: 209, 210
 ar: 145, 146
 as: 145, 146
 ata_id: 226, 226
 autoconf: 183, 183
 autoheader: 183, 183
 autom4te: 183, 183
 automake: 184, 184
 automake-1.12: 184, 184
 autopoint: 195, 195
 autoreconf: 183, 183
 autoscan: 183, 183
 autoupdate: 183, 183
 awk: 192, 192
 badblocks: 161, 162
 base64: 167, 168
 basename: 167, 168
 bash: 186, 186
 bashbug: 186, 186
 bc: 187, 187
 bigram: 193, 193
 bison: 174, 174
 blkid: 155, 156
 blockdev: 155, 156
 bootlogd: 220, 221
 bunzip2: 188, 188
 bzipcat: 188, 188
 bzipcmp: 188, 188
 bzdiff: 188, 188
 bzegrep: 188, 189
 bzfgrep: 188, 189
 bzgrep: 188, 189
 bzip2: 188, 189
 bzip2recover: 188, 189
 bzless: 188, 189
 bzipmore: 188, 189
 c++: 148, 149
 c++filt: 145, 146
 c2ph: 179, 180
 cal: 155, 156
 captinfo: 152, 153

cat: 167, 168
 catchsegv: 131, 136
 cc: 148, 149
 cdrom_id: 226, 226
 cfdisk: 155, 156
 chage: 164, 165
 chattr: 161, 162
 chcon: 167, 168
 chepu: 155, 156
 checkmk: 123, 123
 chem: 198, 198
 chfn: 164, 165
 chgpasswd: 164, 165
 chgrp: 167, 168
 chmod: 167, 168
 chown: 167, 168
 chpasswd: 164, 165
 chroot: 167, 168
 chrt: 155, 156
 chsh: 164, 165
 chvt: 204, 204
 cksum: 167, 168
 clear: 152, 153
 clfskernel-[linux-version]: 256, 258
 clockdiff: 203, 203
 cloog: 143, 143
 cmp: 190, 190
 code: 193, 193
 col: 155, 156
 colcrt: 155, 156
 collect: 226, 226
 colrm: 155, 156
 column: 155, 156
 comm: 167, 168
 compile: 184, 184
 compile_et: 161, 162
 config.charset: 195, 195
 config.guess: 184, 184
 config.rpath: 195, 195
 config.sub: 184, 184
 config_data: 179, 180
 corelist: 179, 180
 cp: 167, 168
 cpan: 179, 180
 cpan2dist: 179, 180
 cpanp: 179, 180
 cpanp-run-perl: 179, 180
 cpp: 148, 149
 create_floppy_devices: 226, 227
 csplit: 167, 169
 ctrlaltdel: 155, 156
 ctstat: 177, 177
 cut: 167, 169
 cytune: 155, 156
 date: 167, 169
 dc: 187, 187
 dd: 167, 169
 ddate: 155, 156
 deallocvt: 204, 204
 debugfs: 161, 162
 delpart: 155, 156
 depcomp: 184, 184
 depmod: 211, 211
 df: 167, 169
 diff: 190, 190
 diff3: 190, 190
 dir: 167, 169
 dircolors: 167, 169
 dirname: 167, 169
 dmesg: 155, 156, 155, 156
 du: 167, 169
 dumpe2fs: 161, 162
 dumpkeys: 204, 204
 e2freefrag: 161, 162
 e2fsck: 161, 162
 e2image: 161, 162
 e2initrd_helper: 161, 162
 e2label: 161, 162
 e2undo: 161, 162
 e4defrag: 161, 162
 echo: 167, 169
 edd_id: 226, 227
 efm_filter.pl: 228, 229
 efm_perl.pl: 228, 229
 egrep: 197, 197
 elfedit: 145, 146
 elisp-comp: 184, 184
 enc2xs: 179, 180
 env: 167, 169
 envsubst: 195, 195
 eqn: 198, 198
 eqn2graph: 198, 198
 ex: 228, 229
 expand: 167, 169

expect: 121, 121
expiry: 164, 165
expr: 167, 169
factor: 167, 169
faillog: 164, 165
fallocate: 155, 156
false: 167, 169
fdformat: 155, 156
fdisk: 155, 156
fgconsole: 204, 204
fgrep: 197, 197
file: 191, 191
filefrag: 161, 162
find: 193, 193
find2perl: 179, 180
findfs: 155, 156
findmnt: 155, 156
firmware.sh: 226, 227
flex: 176, 176
flex++: 176, 176
flock: 155, 156
fmt: 167, 169
fold: 167, 169
frcode: 193, 193
free: 159, 159
fsck: 155, 156
fsck.cramfs: 155, 156
fsck.ext2: 161, 162
fsck.ext3: 161, 162
fsck.ext4: 161, 162
fsck.ext4dev: 161, 162
fsck.minix: 155, 156
fsfreeze: 155, 156
fstab-decode: 220, 221
fstab_import: 226, 227
fstrim: 155, 156
fuser: 214, 214
g++: 148, 149
gawk: 192, 192
gawk-4.1.0: 192, 192
gcc: 148, 149
gcov: 148, 149
gdiffmk: 198, 198
gencat: 131, 136
genl: 177, 177
geqn: 198, 198
getconf: 131, 136
getent: 131, 136
getkeycodes: 204, 204
getopt: 155, 156
gettext: 195, 195
gettext.sh: 195, 195
gettextize: 195, 195
gpasswd: 164, 165
gprof: 145, 146
grap2graph: 198, 198
grcat: 192, 192
grep: 197, 197
grn: 198, 199
grodvi: 198, 199
groff: 198, 199
groffer: 198, 199
grog: 198, 199
grolbp: 198, 199
grolj4: 198, 199
grops: 198, 199
grotty: 198, 199
groupadd: 164, 166
groupdel: 164, 166
groupmems: 164, 166
groupmod: 164, 166
groups: 167, 169
grpck: 164, 166
grpconv: 164, 166
grpunconv: 164, 166
gtbl: 198, 199
gunzip: 202, 202
gzexe: 202, 202
gzip: 202, 202
h2ph: 179, 180
h2xs: 179, 180
halt: 220, 221
hattrib: 231, 231
hcd: 231, 231
hcopy: 231, 231
hdel: 231, 231
hdir: 231, 231
head: 167, 169
hexdump: 155, 156
hformat: 231, 231
hfsutils: 231, 231
hls: 231, 231
hmkdir: 231, 231
hmount: 231, 231

hostid: 167, 169
 hostname: 167, 169
 hostname: 195, 195
 hpftodit: 198, 199
 hpwd: 231, 231
 hrename: 231, 231
 hrmdir: 231, 231
 humount: 231, 231
 hvol: 231, 231
 hwclock: 155, 156
 iconv: 131, 136
 iconvconfig: 131, 136
 id: 167, 169
 ifcfg: 177, 177
 ifnames: 183, 183
 ifstat: 177, 178
 igawk: 192, 192
 indxbib: 198, 199
 info: 224, 224
 infocmp: 152, 153
 infokey: 224, 224
 infotocap: 152, 153
 init: 220, 221
 insmod: 211, 211
 install: 167, 169
 install-info: 224, 224
 install-sh: 184, 184
 instmodsh: 179, 180
 ionice: 155, 156
 ip: 177, 178
 ipcmk: 155, 156
 ipcrm: 155, 157
 ipcs: 155, 157
 isosize: 155, 157
 join: 167, 169
 json_pp: 179, 180
 kbdfinfo: 204, 204
 kbdrate: 204, 204
 kbd_mode: 204, 204
 kill: 155, 157
 killall: 214, 214
 killall5: 220, 221
 kmod: 211, 211
 last: 220, 222
 lastb: 220, 222
 lastlog: 164, 166
 ld: 145, 146
 ld.bfd: 145, 146
 ldattach: 155, 157
 ldconfig: 131, 136
 ldd: 131, 136
 lddlibc4: 131, 136
 less: 201, 201
 less.sh: 228, 230
 lessecho: 201, 201
 lesskey: 201, 201
 lex: 176, 176
 libee-convert: 216, 216
 libnetcfg: 179, 180
 libtool: 175, 175
 libtoolize: 175, 175
 link: 167, 169
 lkbib: 198, 199
 ln: 167, 169
 lnstat: 177, 178
 loadkeys: 204, 204
 loadunimap: 204, 204
 locale: 131, 136
 localedef: 131, 136
 locate: 193, 193
 logger: 155, 157
 login: 164, 166
 logname: 167, 169
 logoutd: 164, 166
 logsave: 161, 162
 look: 155, 157
 lookbib: 198, 199
 losetup: 155, 157
 ls: 167, 169
 lsattr: 161, 162
 lsblk: 155, 157
 lscpu: 155, 157
 lslocks: 155, 157
 lsmod: 211, 212
 lzcat: 207, 207
 lzcmp: 207, 207
 lzdiff: 207, 207
 lzegrep: 207, 207
 lzfgrep: 207, 207
 lzgrep: 207, 207
 lzless: 207, 207
 lzma: 207, 207
 lzmadec: 207, 207
 lzmore: 207, 207

m4: 173, 173
make: 206, 206
makedb: 131, 136
makeinfo: 224, 224
makewhatis: 209, 210
man: 209, 210
man2dvi: 209, 210
man2html: 209, 210
mapscrn: 204, 205
mcookie: 155, 157
md5sum: 167, 169
mdate-sh: 184, 184
mesg: 220, 222
missing: 184, 184
mkdir: 167, 169
mke2fs: 161, 162
mkfifo: 167, 169
mkfs: 155, 157
mkfs.bfs: 155, 157
mkfs.cramfs: 155, 157
mkfs.ext2: 161, 162
mkfs.ext3: 161, 162
mkfs.ext4: 161, 162
mkfs.ext4dev: 161, 162
mkfs.minix: 155, 157
mkinstalldirs: 184, 185
mklost+found: 161, 162
mknod: 167, 169
mkofboot: 234, 234
mkswap: 155, 157
mktemp: 167, 169
mk_cmds: 161, 162
mmroff: 198, 199
modinfo: 211, 212
modprobe: 211, 212
more: 155, 157
mount: 155, 157
mountpoint: 155, 157
msgattrib: 195, 195
msgcat: 195, 195
msgcmp: 195, 195
msgcomm: 195, 196
msgconv: 195, 196
msgen: 195, 196
msgexec: 195, 196
msgfilter: 195, 196
msgfmt: 195, 196
msggrep: 195, 196
msginit: 195, 196
msgmerge: 195, 196
msgunfmt: 195, 196
msguniq: 195, 196
mtrace: 131, 136
mv: 167, 170
mve.awk: 228, 230
namei: 155, 157
ncursesw5-config: 152, 153
neqn: 198, 199
newgrp: 164, 166
newusers: 164, 166
nggettext: 195, 196
nice: 167, 170
nl: 167, 170
nm: 145, 146
nohup: 167, 170
nologin: 164, 166
nproc: 167, 170
nroff: 198, 199
nscd: 131, 136
nstat: 177, 178
nvsetenv: 233, 233
nvsetvol: 233, 233
objcopy: 145, 146
objdump: 145, 146
od: 167, 170
ofboot: 234, 234
ofpath: 234, 234
oldfind: 193, 193
openvt: 204, 205
partx: 155, 157
passwd: 164, 166
paste: 167, 170
patch: 213, 213
pathchk: 167, 170
path_id: 226, 227
pcprofiledump: 131, 136
pdfroff: 198, 199
pdftexi2dvi: 224, 224
peekfd: 214, 214
perl: 179, 180
perl5.18.1: 179, 180
perlbug: 179, 180
perldoc: 179, 181
perlivp: 179, 181

perlthanks: 179, 181
 pfbtops: 198, 199
 pg: 155, 157
 pgawk: 192, 192
 pgawk-4.1.0: 192, 192
 pgrep: 159, 159
 pic: 198, 199
 pic2graph: 198, 199
 piconv: 179, 181
 pidof: 220, 222
 ping: 203, 203
 pinky: 167, 170
 pivot_root: 155, 157
 pkg-config: 154, 154
 pkill: 159, 159
 pl2pm: 179, 181
 pldd: 131, 136
 pltags.pl: 228, 230
 pmap: 159, 159
 pod2html: 179, 181
 pod2latex: 179, 181
 pod2man: 179, 181
 pod2text: 179, 181
 pod2usage: 179, 181
 podchecker: 179, 181
 podselect: 179, 181
 post-grohtml: 198, 199
 poweroff: 220, 222
 pr: 167, 170
 pre-grohtml: 198, 199
 preconv: 198, 199
 printenv: 167, 170
 printf: 167, 170
 prlimit: 155, 157
 prove: 179, 181
 prtstat: 214, 214
 ps: 159, 159
 psed: 179, 181
 psfaddtable: 204, 205
 psfgettable: 204, 205
 psfstriptide: 204, 205
 psfxtable: 204, 205
 pstree: 214, 214
 pstree.x11: 214, 214
 pstruct: 179, 181
 ptar: 179, 181
 ptardiff: 179, 181
 ptargrep: 179, 181
 ptx: 167, 170
 pwcat: 192, 192
 pwck: 164, 166
 pwconv: 164, 166
 pwd: 167, 170
 pwdx: 159, 159
 pwunconv: 164, 166
 py-compile: 184, 185
 ranlib: 145, 146
 raw: 155, 157
 rdisc: 203, 203
 readelf: 145, 146
 readlink: 167, 170
 readprofile: 155, 157
 realpath: 167, 170
 reboot: 220, 222
 recode-sr-latin: 195, 196
 ref: 228, 230
 refer: 198, 199
 rename: 155, 157
 renice: 155, 157
 reset: 152, 153
 resize2fs: 161, 162
 resizecons: 204, 205
 resizepart: 155, 157
 rev: 155, 157
 rm: 167, 170
 rmdir: 167, 170
 rmmmod: 211, 212
 rmt: 223, 223
 roff2dvi: 198, 199
 roff2html: 198, 199
 roff2pdf: 198, 199
 roff2ps: 198, 200
 roff2text: 198, 200
 roff2x: 198, 200
 routef: 177, 178
 routel: 177, 178
 rpcgen: 131, 136
 rsyslogd: 217, 219
 rtacct: 177, 178
 rtcwake: 155, 157
 rtmon: 177, 178
 rtpr: 177, 178
 rtstat: 177, 178
 runcon: 167, 170

runlevel: 220, 222
 runtest: 122, 122
 rview: 228, 230
 rvim: 228, 230
 s2p: 179, 181
 script: 155, 157
 scriptreplay: 155, 157
 scsi_id: 226, 227
 sdiff: 190, 190
 sed: 151, 151
 seq: 167, 170
 setarch: 155, 157
 setfont: 204, 205
 setkeycodes: 204, 205
 settled: 204, 205
 setmetamode: 204, 205
 setsid: 155, 157
 setterm: 155, 158
 setvtrgb: 204, 205
 sfdisk: 155, 158
 sg: 164, 166
 sh: 186, 186
 sha1sum: 167, 170
 sha224sum: 167, 170
 sha256sum: 167, 170
 sha384sum: 167, 170
 sha512sum: 167, 170
 shasum: 179, 181
 showconsolefont: 204, 205
 showkey: 204, 205
 shred: 167, 170
 shtags.pl: 228, 230
 shuf: 167, 170
 shutdown: 220, 222
 size: 145, 146
 skill: 159, 159
 slabtop: 159, 159
 sleep: 167, 170
 sln: 131, 136
 snice: 159, 159
 soelim: 198, 200
 sort: 167, 170
 sotruss: 131, 136
 splain: 179, 181
 split: 167, 170
 sprof: 131, 136
 ss: 177, 178
 stat: 167, 170
 stdbuf: 167, 170
 strings: 145, 147
 strip: 145, 147
 stty: 167, 170
 su: 164, 166
 sulogin: 155, 158
 sum: 167, 170
 swapon: 155, 158
 swappoff: 155, 158
 swapoff: 155, 158
 switch_root: 155, 158
 symlink-tree: 184, 185
 sync: 167, 170
 sysctl: 159, 159
 tabs: 152, 153
 tac: 167, 171
 tail: 167, 171
 tailf: 155, 158
 tar: 223, 223
 taskset: 155, 158
 tbl: 198, 200
 tc: 177, 178
 tcsh: 120, 120
 tcsh-version,: 120, 120
 tcltags: 228, 230
 tee: 167, 171
 telinit: 220, 222
 test: 167, 171
 texi2dvi: 224, 224
 texi2pdf: 224, 225
 texindex: 224, 225
 tfmtodit: 198, 200
 tic: 152, 153
 timeout: 167, 171
 tload: 159, 159
 toe: 152, 153
 top: 159, 160
 touch: 167, 171
 tput: 152, 153
 tr: 167, 171
 tracepath: 203, 203
 tracepath6: 203, 203
 traceroute6: 203, 203
 troff: 198, 200
 true: 167, 171
 truncate: 167, 171

tset: 152, 153
 tsort: 167, 171
 tty: 167, 171
 tune2fs: 161, 162
 tunelp: 155, 158
 tzselect: 131, 136
 udevadm: 226, 226
 udevd: 226, 226
 ul: 155, 158
 umount: 155, 158
 uname: 167, 171
 uncompress: 202, 202
 unexpand: 167, 171
 unicode_start: 204, 205
 unicode_stop: 204, 205
 uniq: 167, 171
 unlink: 167, 171
 unlzma: 207, 207
 unshare: 155, 158
 unxz: 207, 207
 updatedb: 193, 194
 uptime: 159, 160
 usb_id: 226, 227
 useradd: 164, 166
 userdel: 164, 166
 usermod: 164, 166
 users: 167, 171
 utmpdump: 155, 158
 uuid: 155, 158
 uuidgen: 155, 158, 155, 158
 v4l_id: 226, 227
 vdir: 167, 171
 vi: 228, 230
 view: 228, 230
 vigr: 164, 166
 vim: 228, 230
 vim132: 228, 230
 vim2html.pl: 228, 230
 vimdiff: 228, 230
 vimmm: 228, 230
 vimspell.sh: 228, 230
 vimtutor: 228, 230
 vipw: 164, 166
 vmstat: 159, 160
 w: 159, 160
 wall: 155, 158
 watch: 159, 160

wc: 167, 171
 whatis: 209, 210
 whereis: 155, 158
 who: 167, 171
 whoami: 167, 171
 wipefs: 155, 158
 write: 155, 158
 write_cd_rules: 226, 227
 write_net_rules: 226, 227
 xargs: 193, 194
 xgettext: 195, 196
 xsubpp: 179, 181
 xtrace: 131, 137
 xxd: 228, 230
 xz: 207, 207
 xzcat: 207, 208
 xzdec: 207, 208
 yaboot: 234, 234
 yabootconfig: 234, 234
 yacc: 174, 174
 ybin: 234, 234
 yes: 167, 171
 ylwrap: 184, 185
 zcat: 202, 202
 zcmp: 202, 202
 zdiff: 202, 202
 zdump: 131, 137
 zegrep: 202, 202
 zfgrep: 202, 202
 zforce: 202, 202
 zgrep: 202, 202
 zic: 131, 137
 zipdetails: 179, 181
 zless: 202, 202
 zmore: 202, 202
 znew: 202, 202
 zsoelim: 198, 200

Libraries

ld.so: 131, 137
 libanl: 131, 137
 libasprintf: 195, 196
 libbfd: 145, 147
 libblkid: 155, 158
 libBrokenLocale: 131, 137
 libbsd-compat: 131, 137
 libbz2*: 188, 189

libc: 131, 137
 libcheck.{a,so};: 123, 123
 libcidn: 131, 137
 libcloog-isl: 143, 143
 libcom_err: 161, 162
 libcrypt: 131, 137
 libcursesw: 152, 153
 libdl: 131, 137
 libe2p: 161, 162
 libee: 216, 216
 libestr: 215, 215
 libexpect-5.43: 121, 121
 libext2fs: 161, 162
 libfl.a: 176, 176, 176, 176
 libformw: 152, 153
 libg: 131, 137
 libgcc*: 148, 149
 libgcov: 148, 149
 libgettextlib: 195, 196
 libgettextpo: 195, 196
 libgettextsrc: 195, 196
 libgmp: 139, 139
 libgmpxx: 139, 139
 libgomp: 148, 149
 libhistory: 182, 182
 libiberty: 145, 147
 libieee: 131, 137
 libisl: 142, 142
 libltdl: 175, 175
 liblzma: 207, 208
 libm: 131, 137
 libmagic: 191, 191
 libmcheck: 131, 137
 libmemusage: 131, 137
 libmenuw: 152, 153
 libmount: 155, 158
 libmp: 139, 139
 libmpc: 141, 141
 libmpfr: 140, 140
 libmudflap*: 148, 150
 libncursesw: 152, 153
 libnsl: 131, 137
 libnss: 131, 137
 libopcodes: 145, 147
 libpanelw: 152, 153
 libparted: 232, 232
 libpcprofile: 131, 137
 libproc: 159, 160
 libpthread: 131, 137
 libquota: 161, 163
 libreadline: 182, 182
 libresolv: 131, 137
 librpcsvc: 131, 137
 librt: 131, 137
 libSegFault: 131, 137
 libss: 161, 163
 libssp*: 148, 150
 libstdbuf: 167, 171
 libstdc++: 148, 150
 libsupc++: 148, 150
 libtcl-version.so: 120, 120
 libtclstub-version.a: 120, 120
 libthread_db: 131, 137
 libudev: 226, 227
 libutil: 131, 137
 libuuid: 155, 158
 liby.a: 174, 174
 libz: 144, 144
 preloadable_libintl.so: 195, 196

Scripts

checkfs: 237, 237
 cleanfs: 237, 237
 console: 237, 237
 configuring: 240
 udev: 237, 238
 functions: 237, 237
 halt: 237, 237
 ifdown: 237, 237
 ifup: 237, 237
 localnet: 237, 237
 /etc/hosts: 250
 configuring: 250
 mountfs: 237, 237
 mountkernfs: 237, 237
 network: 237, 237
 /etc/hosts: 250
 configuring: 251
 rc: 237, 237
 reboot: 237, 237
 sendsignals: 237, 237
 setclock: 237, 237
 configuring: 240
 static: 237, 237

swap: 237, 237
sysklogd: 237, 238
template: 237, 238

Others

/boot/config-[linux-version]: 256, 257
/boot/System.map-[linux-version]: 256, 258
/dev/*: 107, 117
/etc/clfs-release: 262
/etc/fstab: 104, 255
/etc/group: 96, 115
/etc/hosts: 250
/etc/inittab: 92, 220
/etc/inputrc: 248
/etc/ld.so.conf: 135
/etc/localtime: 133
/etc/login.defs: 164
/etc/nsswitch.conf: 133
/etc/passwd: 96, 115
/etc/profile: 246, 246
/etc/protocols: 172
/etc/resolv.conf: 251
/etc/rsyslog.conf: 218
/etc/services: 172
/etc/udev: 226, 227
/etc/vimrc: 229
/lib/udev: 226, 227
/usr/include/{asm,linux}/*.h: 129, 129
/var/log/btmp: 96, 115
/var/log/lastlog: 96, 115
/var/log/wtmp: 96, 115
/var/run/utmp: 96, 115
dhcpcd: 253
man pages: 130, 130
parted: 232, 232
partprobe: 232, 232
yaboot.conf: 234, 234