

Cross-Compiled Linux From Scratch

Version 1.1.0-x86_64-Pure64

**Jim Gifford
Ryan Oliver**

Cross-Compiled Linux From Scratch: Version 1.1.0-x86_64-Pure64

by Jim Gifford and Ryan Oliver

Copyright © 2005–2008 Jim Gifford & Ryan Oliver

Based on LFS, Copyright © 1999–2006 Gerard Beekmans

Copyright © 2005–2008, 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 (c) 1999–2006, 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	viii
i. Foreword	viii
ii. Audience	viii
iii. Prerequisites	ix
iv. Host System Requirements	x
v. Typography	xi
vi. Structure	xii
vii. Errata	xiii
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 x86_64-64	15
1.5. Resources	16
1.6. Help	17
II. Preparing for the Build	19
2. Preparing a New Partition	20
2.1. Introduction	20
2.2. Creating a New Partition	20
2.3. Creating a File System on the Partition	20
2.4. Mounting the New Partition	21
3. Packages and Patches	23
3.1. Introduction	23
3.2. All Packages	23
3.3. Additional Packages for x86_64	28
3.4. Needed Patches	29
3.5. Additional Patches for x86_64	31
4. Final Preparations	32
4.1. About $\{\text{CLFS}\}$	32
4.2. Creating the $\{\text{CLFS}\}/\text{tools}$ Directory	32
4.3. Creating the $\{\text{CLFS}\}/\text{cross-tools}$ Directory	33
4.4. Adding the CLFS User	33
4.5. Setting Up the Environment	34
4.6. About the Test Suites	35
III. Make the Cross-Compile Tools	36
5. Constructing Cross-Compile Tools	37
5.1. Introduction	37
5.2. Build CFLAGS	37
5.3. Build Flags	37
5.4. Build Variables	38
5.5. Linux-Headers-2.6.24.7	39
5.6. File-4.23	40
5.7. Cross Binutils-2.18	41
5.8. Cross GCC-4.2.4 - Static	43

5.9. Glibc-2.7	45
5.10. Cross GCC-4.2.4 - Final	47
IV. Building the Basic Tools	49
6. Constructing a Temporary System	50
6.1. Introduction	50
6.2. Build Variables	50
6.3. Binutils-2.18	51
6.4. GCC-4.2.4	52
6.5. Ncurses-5.6	54
6.6. Bash-3.2	55
6.7. Bzip2-1.0.4	57
6.8. Coreutils-6.9	58
6.9. Diffutils-2.8.7	59
6.10. Findutils-4.2.32	60
6.11. Zlib-1.2.3	61
6.12. File-4.23	62
6.13. Gawk-3.1.6	63
6.14. Gettext-0.17	64
6.15. Grep-2.5.3	65
6.16. Gzip-1.3.12	66
6.17. Make-3.81	67
6.18. Patch-2.5.9	68
6.19. Sed-4.1.5	69
6.20. Tar-1.20	70
6.21. Texinfo-4.11	71
6.22. To Boot or to Chroot?	72
7. If You Are Going to Boot	73
7.1. Introduction	73
7.2. Creating Directories	73
7.3. Creating Essential Symlinks	74
7.4. E2fsprogs-1.40.4	75
7.5. Sysvinit-2.86	76
7.6. Module-Init-Tools-3.4	78
7.7. Util-linux-ng-2.14	79
7.8. Udev-124	80
7.9. Creating the passwd, group, and log Files	81
7.10. Linux-2.6.24.7	84
7.11. Building a bootloader	86
7.12. Bin86-0.16.17	87
7.13. Lilo-22.8	88
7.14. Setting Up the Environment	89
7.15. Build Flags	89
7.16. Creating the /etc/fstab File	89
7.17. CLFS-Bootscripts-1.1.0	91
7.18. Populating /dev	92
7.19. Changing Ownership	92
7.20. Making the CLFS System Bootable	92

7.21. What to do next	94
8. If You Are Going to Chroot	95
8.1. Introduction	95
8.2. E2fsprogs-1.40.4	96
8.3. Util-linux-ng-2.14	97
8.4. Mounting Virtual Kernel File Systems	98
8.5. Entering the Chroot Environment	98
8.6. Changing Ownership	99
8.7. Creating Directories	100
8.8. Creating Essential Symlinks	100
8.9. Build Flags	101
8.10. Creating the passwd, group, and log Files	101
8.11. Mounting Kernel Filesystems	103
V. Building the CLFS System	104
9. Constructing Testsuite Tools	105
9.1. Introduction	105
9.2. Tcl-8.4.16	106
9.3. Expect-5.43.0	107
9.4. DejaGNU-1.4.4	108
9.5. Tree-1.5.1.1	109
10. Installing Basic System Software	110
10.1. Introduction	110
10.2. Package Management	110
10.3. About Test Suites, Again	113
10.4. Temporary Perl-5.8.8	114
10.5. Linux-Headers-2.6.24.7	115
10.6. Man-pages-3.01	116
10.7. Glibc-2.7	117
10.8. Adjusting the Toolchain	123
10.9. Binutils-2.18	124
10.10. GCC-4.2.4	127
10.11. Sed-4.1.5	129
10.12. E2fsprogs-1.40.4	130
10.13. Coreutils-6.9	133
10.14. Iana-Etc-2.20	138
10.15. M4-1.4.10	139
10.16. Bison-2.3	140
10.17. Ncurses-5.6	141
10.18. Procps-3.2.7	143
10.19. Libtool-1.5.26	145
10.20. Flex-2.5.35	146
10.21. IPRoute2-2.6.23	147
10.22. Perl-5.8.8	149
10.23. Readline-5.2	152
10.24. Zlib-1.2.3	153
10.25. Autoconf-2.61	154
10.26. Automake-1.10.1	155

10.27. Bash-3.2	157
10.28. Bzip2-1.0.4	159
10.29. Diffutils-2.8.7	161
10.30. File-4.23	162
10.31. Findutils-4.2.32	163
10.32. Gawk-3.1.6	165
10.33. Gettext-0.17	166
10.34. Grep-2.5.3	168
10.35. Groff-1.19.2	169
10.36. Gzip-1.3.12	172
10.37. Inetutils-1.5	174
10.38. Kbd-1.13	176
10.39. Less-418	178
10.40. Make-3.81	179
10.41. Man-1.6e	180
10.42. Mktmp-1.5	182
10.43. Module-Init-Tools-3.4	183
10.44. Patch-2.5.9	185
10.45. Psmisc-22.6	186
10.46. Shadow-4.1.2	187
10.47. Sysklogd-1.5	190
10.48. Sysvinit-2.86	191
10.49. Tar-1.20	194
10.50. Texinfo-4.11	195
10.51. Udev-124	196
10.52. Util-linux-ng-2.14	198
10.53. Vim-7.1	201
10.54. Bin86-0.16.17	204
10.55. Lilo-22.8	205
10.56. About Debugging Symbols	206
10.57. Stripping	206
11. Setting Up System Bootscripts	207
11.1. Introduction	207
11.2. CLFS-Bootscripts-1.1.0	208
11.3. How Do These Bootscripts Work?	209
11.4. Device and Module Handling on a CLFS System	210
11.5. Configuring the setclock Script	213
11.6. Configuring the Linux Console	214
11.7. Configuring the syslogd script	215
11.8. Creating the /etc/inputrc File	215
11.9. The Bash Shell Startup Files	216
11.10. Configuring the localnet Script	218
11.11. Customizing the /etc/hosts File	218
11.12. Creating custom symlinks to devices	219
11.13. Configuring the network Script	221
12. Making the CLFS System Bootable	223
12.1. Introduction	223

12.2. Creating the /etc/fstab File	223
12.3. Linux-2.6.24.7	224
12.4. Making the CLFS System Bootable	227
13. The End	230
13.1. The End	230
13.2. Get Counted	230
13.3. Rebooting the System	230
13.4. What Now?	231
VI. Appendices	233
A. Acronyms and Terms	234
B. Acknowledgments	237
C. Dependencies	240
D. x86 Dependencies	247
E. Open Publication License	248
Index	251

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)

Jim Gifford, CLFS Project Co-leader

Ryan Oliver, CLFS Project Co-leader

Joe Ciccone, Justin Knierim, Chris Staub, Matt Darcy, Ken Moffat,

Manuel Canales Esparcia, and Nathan Coulson - 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/essential_prereading.txt

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 2.18 are not recommended as they have not been tested)
- **Bison-1.875**
- **Bzip2-1.0.2**
- **Coreutils-5.0** (or Sh-Utills-2.0, Textutils-2.0, and Fileutils-4.1)
- **Diffutils-2.8**
- **Findutils-4.1.20**
- **Gawk-3.0**
- **Gcc-2.95.3** (Versions greater than 4.2.4 are not recommended as they have not been tested)
- **Glibc-2.2.5** (Versions greater than 2.7 are not recommended as they have not been tested)
- **Grep-2.5**
- **Gzip-1.2.4**
- **Make-3.79.1**
- **Ncurses-5.3**
- **Patch-2.5.4**
- **Sed-3.0.2**
- **Tar-1.14**
- **Texinfo-4.4**

To see whether your host system has all the appropriate versions, run the following:

```
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
/lib/libc.so.6 | head -n1 | cut -d" " -f1-7
grep --version | head -n1
gzip --version | head -n1
make --version | head -n1
tic -V
patch --version | head -n1
sed --version | head -n1
tar --version | head -n1
makeinfo --version | head -n1

EOF

bash version-check.sh
```

Typography

To make things easier to follow, there are a few typographical conventions used throughout this book. This section contains some examples of the typographical format found throughout 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; Appendix B gives acknowledgments to people who have helped work on the CLFS project and website; Appendix C and D have information about package dependencies and the 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/clfs/errata/1.1.0/> 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.
- Jeremy Utley - Release Manager 1.x Series.

Our CLFS Team:

- Joe Ciccone - Alpha, MIPS, Sparc builds.
- Nathan Coulson - Bootscripts.
- Matt Darcy - x86, X86_64, and Sparc builds.
- Manuel Canales Esparcia - Book XML.
- Karen McGuinness - Proofreader.
- 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.

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

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 the Linux From Scratch LiveCD. This CD works well as a host system, providing all the tools you need to successfully follow the instructions in this book. It does also contain source packages and patches for the LFS book, and a copy of the LFS book, but not the needed packages or book for CLFS. You can still use the CD for building CLFS, but you will need to download the packages, patches and book separately. You can also look at <http://hints.cross-lfs.org/lfscd-remastering-howto.txt> for information on building your own CD, replacing the LFS packages and book with those for CLFS. Once you have the CD, no network connection or additional downloads are necessary. For more information about the LFS LiveCD or to download a copy, visit <http://www.linuxfromscratch.org/livecd/>.

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 1.1.0 of the Cross-Compiled Linux From Scratch book, dated September 14, 2008. 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:

- September 13, 2008
 - [jiccone] - Checked and updated all download locations.
- September 7, 2008
 - [jiccone] - A lot of text changes.
- August 27, 2008
 - [jiccone] - Add Texinfo to the list of Host System Requirements.
- August 24, 2008
 - [jiccone] - Update the paths to ld.so when configuring Glibc for each arch.
- July 13, 2008
 - [jiccone] - Add a page right before Entering the Chroot Environment. This page covers the problem of building a 32bit system from a 64bit host or building a older 32bit system from another newer 32bit system.
- July 12, 2008
 - [jiccone] - Drop the minix tools from the chroot Util-Linux-NG build.
- July 11, 2008
 - [ken] - Add a patch to address known perl vulnerabilities.
- July 7, 2008
 - [jiccone] - Change the command that creates the clfs user to not include -k /dev/null. Newer versions of Shadow (4.1.2) apparently any argument passed to -k to be a directory.
- July 7, 2008
 - [jiccone] - Fix a omission in the Udev lib64 patch. Thanks to AcidPoison for catching this and reporting it in Trac.
- July 6, 2008
 - [jiccone] - Now that shadow doesn't provide any libraries we only need to build it once on multilib.
- June 28, 2008
 - [jiccone] - Downgraded the Vim Branch Update patch back to -2 from -3.
 - [jiccone] - Remove the commands that relocate the Shadow libraries as they are not provided in newer versions of shadow. Also only build 64bit, since there are no libraries anymore.
 - [jiccone] - Link /tools/bin/echo to /bin/echo for one of the Glibc tests.

- [jccicone] - Updated GCC to 4.2.4.
- June 28, 2008
 - [jccicone] - Fixed the Glibc i586 chk patch entity.
 - [jccicone] - Updated Tar to 1.20.
 - [jccicone] - Updated Udev to 124.
 - [jccicone] - Updated the Ncurses Branch Update patch to -3/
 - [jccicone] - Add a config.cache entry to Coreutils so that the **df** program will be built
 - [jccicone] - Fixed descriptions for lib64= in procps on the various multilib pages.
 - [jccicone] - Corrected the entities for the lilo x86_64 cross compile patch.
 - [jccicone] - Added a patch to perl so it wont include asm/page.h.
 - [jccicone] - Updated the Vim Branch Update patch to -3.
 - [jccicone] - Updated Shadow to 4.1.2.
 - [jccicone] - Fixed GCC Posix patch MD5SUM.
 - [jccicone] - Fixed Inetutils MD5SUM.
 - [jccicone] - Updated Util-Linux-NG to 2.14.
 - [jccicone] - Updated the Util-Linux-NG homepage link.
 - [jccicone] - Updated Man-Pages to 3.01.
 - [jccicone] - Updated the Bash Fixes patch to -8.
- June 17, 2008
 - [Ken] - Fix all known kernel vulnerabilities by upgrading to 2.6.24.7 plus a patch extracted from debian for the more recent fixes.
- June 7, 2008
 - [Chris] - Updated dependency info for several packages.
- Date
 - [jccicone] - Updated Flex to 2.5.35. This update fixes numerous compilation issues against 2.5.34.
- February 12, 2008
 - [ken] - Added cs_CZ.UTF-8 locale for grep's fmbtest.sh test.
- Frbruary 3, 2008
 - [jccicone] - Updated the Kernel to 2.6.24.
- Feburary 2, 2008
 - [jccicone] - Updated Man-Pages to 2.77.
 - [jccicone] - Updated Libtool to 1.5.26.
 - [jccicone] - Updated Findutils to 4.2.32.
 - [jccicone] - Updated Automake to 1.10.1.
 - [jccicone] - Updated the bootscripts to 1.1-pre10. These use udevadm instead of the individual udevcontrol, udevtrigger, and udevsettle programs. Which are currently symlinked to udevadm.

- January 20, 2008
 - [Bigdissaved] - Changed the minimum `{CLFS}` size from 1.5Gb to 2.5Gb. Thanks to Copper for seeing this
- January 19, 2008
 - [jiccone] - Update the Kernel to 2.6.23.14.
- January 19, 2008
 - [jiccone] - Updated Man-Pages to 2.76.
 - [jiccone] - Updated Util-Linux-NG to 2.13.1.
 - [jiccone] - Updated Less to 418.
- January 14, 2008
 - [ken] - Add patch for man pages in module-init-tools.
- January 13, 2008
 - [jiccone] - Updated the Kernel to 2.6.23.13.
- January 6, 2008
 - [jiccone] - Updated File to 4.23.
 - [ken] - Updated Glibc branch update patch to 1A, so that ppc and ppc64 build again.
- January 2, 2008
 - [jiccone] - Updated E2fsprogs to 1.40.4.
- January 1, 2007
 - [jiccone] - Updated Shadow to 4.1.0.
- December 29, 2007
 - [ken] - Make the current Module-Init-Tools testsuite run, update the instructions because distclean is no longer useful, and comment on the DOCBOOKTOMAN parameter.
- December 23, 2007
 - [jiccone] - Added a config.cache entry to Gettext to resolve some errors.
- December 19, 2007
 - [jiccone] - Updated Util-Linux-NG to 2.13.0.1.
 - [jiccone] - Updated Various Branch Update Patches.
 - [jiccone] - Updated Udev to 118.
 - [jiccone] - Updated Psmisc to 22.6.
 - [jiccone] - Updated Module Init Tools to 3.4.
 - [jiccone] - Updated Man-Pages to 2.73.
 - [jiccone] - Updated the Kernel to 2.6.23.12.
 - [jiccone] - Updated Less to 416.
- December 16, 2007
 - [jiccone] - Updated Gettext to 0.17.
 - [jiccone] - Updated e2fsprogs to 1.40.3.

- [jiccone] - Updated Flex to 2.5.34.
- [jiccone] - Updated the Kernel to 2.6.23.11.
- [jiccone] - Updated the bootscripts to 1.1-pre9.
- December 10, 2007
 - [jiccone] - Added --disable-makeinstall-chown to util-linux-ng in the boot section.
- November 27, 2007
 - [jiccone] - Updated the bootscripts to 1.1-pre8.
- October 30, 2007
 - [jim] - Updated to TCL 8.4.16.
 - [jim] - Updated to Man-Pages 2.67.
 - [jim] - Replaced Shadow Patch with better patch.
- October 29, 2007
 - [jim] - Updated Binutils 2.18 Branch Update Patch.
 - [jim] - Updated Vim 7.1 Branch Update Patch.
 - [jim] - Added Ncurses 5.6 Branch Update Patch.
 - [jim] - Updated to Gawk 3.1.6.
 - [jim] - Updated to Shadow 4.0.18.2.
 - [jim] - Added Updated Useradd Fix Patch to Shadow 4.0.18.2.
- October 26, 2007
 - [jim] - Added Binutils 2.18 Branch Update Patch.
 - [jim] - Added GCC 4.2.2 Branch Update Patch.
 - [jim] - Added Glibc 2.7 Branch Update Patch.
- October 25, 2007
 - [jim] - Updated the bootscripts to 1.1-pre7.
 - [jim] - Updated to Glibc 2.7.
 - [jim] - Updated Bootscripts build to make sure the udev devices get created correctly in boot section.
- October 16, 2007
 - [jim] - Updated the bootscripts to 1.1-pre6. Combined with cblfs bootscripts, all bootscripts in one package.
 - [jim] - Updated to Linux 2.6.23.1.
 - [jim] - Removed Linux-Headers Package.
 - [jim] - Updated to GCC 4.2.2.
 - [jim] - Updated to Tar 1.19.
 - [jim] - Updated to Texinfo 4.11.
 - [jim] - Updated to Less 409.
 - [jim] - Updated to Udev 116.
 - [jim] - Added patch to correct the detection of texinfo in binutils.

- [jim] - Updated to IPRoute2 2.6.23.
- October 13, 2007
 - [jiccone] - Updated the bootscripts to 1.1-pre5. The Makefile now creates the static nodes that the udev Makefile used to.
 - [jim] - Updated the boot section Util-Linux-NG Build. Fixes issues where files were missing during bootup and the ability to login.
- September 24, 2007
 - [Chris] - More dependency updates for several packages.
- September 23, 2007
 - [Chris] - Updates to dependency info for several packages.
- September 12, 2007
 - [Chris] - Added more new programs to Util-Linux-Ng installed programs list.
- September 11, 2007
 - [Chris] - Updated list of installed programs for Util-Linux-ng.
 - [jim] - Updated CLFS Bootscripts to 1.1-pre4.
- September 10, 2007
 - [Chris] - Added needed mkdir command to IPRoute2 instructions.
 - [Chris] - Fixed descriptive text in testsuite-tools introduction page. Reported by gomoko (ticket #118).
- September 09, 2007
 - [jiccone] - Build a PIC libfl.a with Flex.
 - [jim] - Updated Util-Linux-ng in Boot Section, to cross-compile.
- September 03, 2007
 - [jim] - Updated Udev to 115. This change also removed udev-cross-lfs package since our rules are the same as base ones installed.
 - [jim] - Updated Grep to 2.5.3.
 - [jim] - Updated Man-Pages to 2.64.
 - [jim] - Updated TCL to 8.4.15.
 - [jim] - Updated to Linux 2.6.22.6.
 - [jim] - Updated to Linux Headers 2.6.22.6-09032007.
 - [jim] - Updated Less to 406.
 - [jim] - Updated Sysklogd to 1.5.
 - [jim] - Updated Kbd to 1.13.
 - [jim] - Updated IPRoute2 to 2.6.22-070710.
 - [jim] - Updated Shadow Patch for more fixes. Man Page Typos, buffer overflows, useradd -G fixes, better name support, and the previous useradd fix.
 - [jim] - Updated Bash 3.2 Patch to -6.
 - [jim] - Updated Readline 5.2 Patch to -3.

- [jim] - Updated Vim 3.1 Patch to -3.
- September 02, 2007
 - [jim] - Updated Util-linux-ng to 2.13.
 - [jim] - Updated Tree to 1.5.1.1.
 - [jccicone] - Updated Glibc to 2.6.1.
 - [jim] - Updated Binutils to 2.18.
- August 05, 2007
 - [jccicone] - Updated the bash config.cache to contain all of the differences between a cross-compiled and a native build.
- July 23, 2007
 - [jccicone] - Updated Texinfo to 4.9.
 - [jccicone] - Updated E2fsprogs to 1.40.2.
 - [jccicone] - Updated M4 to 1.4.10.
 - [jccicone] - Updated Tar to 1.18.
 - [jccicone] - Updated Libtool to 1.5.24.
 - [jccicone] - Updated the Man Pages to 2.63.
- July 22, 2007
 - [jccicone] - Updated GCC to 4.2.1.
- July 20, 2007
 - [ken] - Make coreutils build in temp-system when the host cannot run the target programs.
- July 1, 2007
 - [jccicone] - Updated the download location for the Man Pages package.
- June 23, 2007
 - [jccicone] - Updated to Glibc 2.6. This includes adding a sed to Coreutils and Gzip to rename their internal implementations of futimens which are incompatible with the implementation provided by Glibc.
- June 18, 2007
 - [jccicone] - Updated to Tar 1.17.
- June 17, 2007
 - [jccicone] - Updated to Vim 7.1.
 - [jccicone] - Updated to Findutils 4.2.31.
 - [jccicone] - Updated to File 4.21.
 - [jccicone] - Updated to Less 403.
 - [jccicone] - Updated to Man Pages 2.57.
- June 16, 2007
 - [Chris] - Removed creation of config.cache from temp-system Coreutils, as it no longer looks for setvbuf.
- May 19, 2007

- [jiccone] - Added a patch to Perl that accounts for a change in GCC 4.2.0. The patch causes makedepend.sh to also remove <command-line> as well as <command line>.
- May 18, 2007
 - [jiccone] - Updated to GCC 4.2.0.
- April 30, 2007
 - [jiccone] - Updated to Man Pages 2.46.
- April 27, 2007
 - [jhuntwork] - We don't use the uname patch for temp-system Coreutils. No need to 'touch man/uname.1'.
- April 24, 2007
 - [jiccone] - Updated to Psmisc 22.5.
- April 20, 2007
 - [jiccone] - Updated to Gzip 1.3.12.
 - [jiccone] - Updated to Psmisc 22.4.
 - [jiccone] - Updated to Man Pages 2.44.
 - [jiccone] - Updated to M4 1.4.9.
 - [jiccone] - Updated to Coreutils 6.9.
- April 10, 2007
 - [Chris] - Updated Gzip's list of installed programs - compress is no longer installed, and gunzip, uncompress, and zcat are now bash scripts rather than links.
- February 25, 2007
 - [jim] - Updated to Linux 2.6.20.1.
 - [jim] - Updated to Linux Headers 2.6.20.1-02252007.
- February 17, 2007
 - [Chris] - Moved the build of the File package in cross-tools to the beginning of the chapter.
 - [jim] - Updated to Gzip 1.3.11.
- February 16, 2007
 - [Chris] - Changed the wording in the "Build Variables" page to make it clearer.
 - [Chris] - Added command to Udev installation to remove Udev rules it installs.
 - [Chris] - Removed Ncurses patch for buggy versions of Bash from the final-system build (it's still in the temp-system).
 - [Chris] - Removed unneeded /tools/bin/cc symlink.
- February 15, 2007
 - [jim] - Fix a building of Ncurses when an unpatched version of Bash 3.x is used.
- February 14, 2007
 - [jim] - Updated to GCC 4.1.2.
- February 5, 2007

- [jim] - Updated to Automake 1.10.
- February 4, 2007
 - [jim] - Updated to Linux 2.6.20.
 - [jim] - Updated to Linux Headers 2.6.20-02042007.
 - [jim] - Updated to Udev 105.
- January 23, 2007
 - [jim] - Updated to Linux Headers 2.6.19.2-01232007.
 - [jim] - Update to Glibc Branch Update Patch.
- January 15, 2007
 - [jim] - Updated to Linux 2.6.19.2.
 - [jim] - Updated to Linux Headers 2.6.19.2-01152007.
 - [jim] - Updated to Tree 1.5.1.
 - [jim] - Updated to Udev 104.
 - [jim] - Updated to Udev Rules 1.1-pre4.
 - [Chris] - Removed obsolete note from bzip2 instructions about performing `rm /usr/bin/bz*`.
- January 12, 2007
 - [jim] - Update to Glibc Branch Update Patch.
 - [jim] - Update to Binutils Branch Update Patch.
- January 7, 2007
 - [jim] - Updated to Bzip2 1.0.4.
- December 23, 2006
 - [jccicone] - Updated to Ncurses 5.6.
- December 15, 2006
 - [jim] - Updated to Gzip 1.3.9.
 - [jim] - Updated to IPRoute2 2.6.19-061214.
- December 13, 2006
 - [jim] - Added File to Cross-tools.
- December 12, 2006
 - [jim] - Updated to Linux 2.6.19.1.
 - [jim] - Updated to Linux Headers 2.6.19.1-12122006.
 - [jim] - Added Readline Upstream Patch.
 - [jim] - Update Bash Upstream Patch.
 - [jim] - Updated to File 4.19.
- December 10, 2006
 - [jim] - Updated to Gzip 1.3.8.

- December 9, 2006
 - [jim] - Updated to Tar 1.16.1.
- December 7, 2006
 - [jim] - Updated to Gzip 1.3.7.
 - [jim] - Updated to Coreutils 6.7.
 - [jim] - Reverting sysroot changes.
- December 1, 2006
 - [jim] - Updated to Man-Pages 2.43.
 - [jim] - Removed MINOR patch and replaced it with a more complete upstream patch with all current updates to the 2.5 branch of glibc.
 - [jim] - Added Binutils Branch Update Patch.
 - [jim] - Reverted change in gettext build in temp-system.
 - [jim] - Updated to Coreutils 6.7.
- November 29, 2006
 - [jim] - Updated to Linux Headers 2.6.19-11302006.
 - [jim] - Updated to Udev Rules 1.1-pre3.
- November 29, 2006
 - [jim] - Updated to Linux 2.6.19.
 - [jim] - Updated to Linux Headers 2.6.19-11292006.
- November 27, 2006
 - [jim] - Updated to Gettext 0.16.1.
- November 26, 2006
 - [jim] - Updated to Man-Pages 2.42.
 - [jim] - Updated to Gzip 1.3.6.
- November 25, 2006
 - [jim] - Updated to Findutils 4.2.29.
- November 23, 2006
 - [Chris] - Added /bin/rm to essential symlinks - e2fsprogs testsuite hard-codes the location to that binary.
- November 22, 2006
 - [jim] - Updated to Coreutils 6.6.
 - [jim] - Updated to Man 1.6e.
- November 20, 2006
 - [jim] - Updated to Coreutils 6.5.
 - [jim] - Added Patch for Coreutils 6.5 ls segfault.
 - [jim] - Updated to M4 1.4.8.
 - [jim] - Updated to Linux 2.6.18.3.

- [jim] - Updated to Linux Headers 2.6.18.3-11202006.
- November 17, 2006
 - [jim] - Updated to Autoconf 2.61.
- November 15, 2006
 - [jim] - Updated to IANA-Etc 2.20.
- November 14, 2006
 - [jim] - Change to E2FSProgs, does not pick up the CC or LD variables.
- November 12, 2006
 - [jim] - Updated to Linux Headers 2.6.18.2-11122006.
 - [jim] - Bash 3.2 patch updated with upstream fixes 001-005.
- November 5, 2006
 - [jim] - Updated to Linux 2.6.18.2.
 - [jim] - Updated to Linux Headers 2.6.18.2-11052006.
- November 4, 2006
 - [Chris] - Modified the instructions for disabling the installation of the groups program to account for changes in shadow's Makefile.
- November 2, 2006
 - [jim] - Updated to File 4.18.
- October 31, 2006
 - [jim] - Bash 3.2 patch updated with upstream fixes.
 - [jim] - Updated to Bootscripts 1.1-pre3.
- October 30, 2006
 - [jim] - Updated to Udev Rules 1.1-pre2.
- October 27, 2006
 - [jim] - Updated to Bootscripts 1.1-pre2.
 - [jim] - Updated syslog.conf file.
- October 26, 2006
 - [jim] - Updated to Linux-Headers 2.6.18.1-10262006.
 - [jim] - Updated to Gettext 0.16.
 - [jim] - The nscd sed is no longer needed to build Glibc 2.5.
- October 25, 2006
 - [Chris] - Updated to Psmisc 22.3.
 - [Chris] - Updated to Tcl 8.4.14.
 - [Chris] - Moved e2fsprogs and sed before coreutils, as Coreutils testsuite now requires e2fsprogs.
- October 23, 2006
 - [Chris] - Removed unneeded additions to config.cache for temp-system Tar and Coreutils.

- [jim] - Added touch man/uname.1 in temp-system Coreutils to prevent a build failure.
- [jim] - Added Patch for a Versioning issue with Glibc 2.5.
- October 22, 2006
 - [jim] - Updated to Linux 2.6.18.1.
 - [jim] - Updated to Linux Headers 2.6.18.1-10222006.
 - [jim] - Updated to Coreutils 6.4.
 - [jim] - Removed out of place udev text in bootscripts Networking.
- October 21, 2006
 - [jim] - Updated to Inetutils 1.5.
 - [jim] - Updated to IPRoute2 2.6.18-061002.
 - [jim] - Updated to Tar 1.16.
 - [jim] - Updated to Udev 103.
 - [jim] - Updated to Coreutils 6.3.
- October 20, 2006
 - [Chris] - Removed command to add "setvbuf_reversed" to temp-system bash as it is no longer needed with Bash 3.2.
 - [Chris] - Added command to man instructions to comment out MANPATH /usr/local/man (in addition to /usr/man).
- October 15, 2006
 - [jim] - Updated Bash 3.2 with some unofficial upstream fixes. Thank you Chet Ramey.
- October 12, 2006
 - [jim] - Added patch to fix build issue with Bash 3.2.
 - [jim] - Updated to Man-Pages 2.41.
 - [jim] - Updated to Glibc 2.5.
- October 11, 2006
 - [jim] - Updated to Bash 3.2.
 - [jim] - Updated to Readline 5.2.
 - [jim] - Updated to Man-Pages 2.40.
 - [jim] - Updated to Udev 102.
- October 4, 2006
 - [manuel] - Changed book license to the Open Publication License v1.0.
- October 1, 2006
 - [Chris] - Removed obsolete command from Tcl instructions, as the issue it fixes has been fixed in the latest Tcl version.
- September 29, 2006
 - [jim] - Updated to Linux Headers to 2.6.18-09302006.

- [jim] - New patch for util-linux, replaces GCC 4 Fixes, the new patches fixes the same issues, and removes the use of syscalls. Syscalls were removed the unistd.h during sanitization.
- September 28, 2006
 - [jim] - Added patch for Util-Linux to fix build issues due to headers change in the kernel.
 - [jim] - Updated to Texinfo 4.8a.
- September 25, 2006
 - [jim] - Updated Bootscripts to 1.1-pre1. Adding in the boot logging feature.
 - [jim] - Updated to Udev 100.
 - [jim] - Updated Udev Rules for Udev 100.
 - [jim] - Updated to M4 1.4.7.
 - [jim] - Updated to Linux to 2.6.18.
 - [jim] - Updated to Linux Headers to 2.6.18-09252006.
 - [jim] - Udev library libvolume_id is used by HAL, which makes us change our build to a multilib build of udev libraries only for 32 and N32.
- September 18, 2006
 - [jim] - Updated Bison to a multilib build in all support architectures.
- August 28, 2006
 - [jim] - Updated to Gettext 0.15.
 - [jim] - Updated to Shadow 4.0.18.1.
 - [jim] - Updated to Coreutils 5.97.
 - [jim] - Updated to Procs 3.2.7.
 - [jim] - Updated to Man-Pages 2.39.
 - [jim] - Updated to Autoconf 2.60.
 - [jim] - Updated to TCL 8.4.13.
 - [jim] - Updated to M4 1.4.6.
 - [jim] - Updated to Findutils 4.2.28.
 - [jim] - Updated to Coreutils Uname Patch -2, which supports all architectures.
 - [jim] - Updated to Vim Patch -10, which contains upstream fixes.
 - [jim] - Updated to Udev 098 and udev-cross-lfs to 1.1-08272006 rules.
- August 28, 2006
 - [jim] - Changelog restarted, see the 1.0.0 book for the old changelog.

Branch Synced from the Release of LFS 6.0 on February 23rd, 2005

1.4. Changelog for x86_64-64

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:

- June 02, 2008
 - [ken] - create libstdc++.so symlinks in /usr/lib64 so that the glibc and ld tests all link, and delete them safely before installing binutils.
 - [ken] - ensure binutils does not install libiberty.a into /usr/lib64.
- January 22, 2008
 - [ken] - Use `#{CLFS}/etc/lilo.conf` when running lilo in 'boot'. Thanks to b-vol for highlighting the issue.
- February 19, 2007
 - [jim] - Updated to Lilo 22.8.
- February 05, 2007
 - [ken] - Target for lilo in boot/ is now lilo.static.
 - [ken] - Remove sed for gettext testsuite, no longer needed.
- November 14, 2006
 - [ken] - Fixed longstanding typo in lilo instructions for boot.
- August 28, 2006
 - [jim] - Updated to Lilo 22.7.3.
- August 28, 2006
 - [jim] - Changelog restarted, see the 1.0.0 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 it's own News Server, but we do provide access via `gmane.org` <http://gmane.org>. If you want to subscribe to the Cross-LFS via a newsreader you can utilize `gmane.org`. You can find the game 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/errors.txt>.

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 1.1.0)
- 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 your 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 2.5 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.

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

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.

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_.2F_Block_Device_.2F_Inode_Management_Tools.

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-1.40.4.tar.bz2
cd e2fsprogs-1.40.4
mkdir build
cd build
../configure
make #note that we intentionally don't 'make install' here!
./misc/mke2fs /dev/[xxx]
cd /tmp
rm -rf e2fsprogs-1.40.4
```

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/1.1.0/>.

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
```

3.2. All Packages

Download or otherwise obtain the following packages:

- **Autoconf (2.61) - 1,024 KB:**

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

Download: <http://ftp.gnu.org/gnu/autoconf/autoconf-2.61.tar.bz2>

MD5 sum: 36d3fe706ad0950f1be10c46a429efe0

- **Automake (1.10.1) - 904 KB:**

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

Download: <http://ftp.gnu.org/gnu/automake/automake-1.10.1.tar.bz2>

MD5 sum: 4510391e6b3edaa4cffb3ced87c9560c

- **Bash (3.2) - 2,476 KB:**

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

Download: <http://ftp.gnu.org/gnu/bash/bash-3.2.tar.gz>

MD5 sum: 00bfa16d58e034e3c2aa27f390390d30

- **Bash Documentation (3.2) - 2,148 KB:**

Download: <http://ftp.gnu.org/gnu/bash/bash-doc-3.2.tar.gz>

MD5 sum: 0e904cb46ca873fcfa65df19b024bec9

- **Binutils (2.18) - 14,671 KB:**

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

Download: <http://ftp.gnu.org/gnu/binutils/binutils-2.18.tar.bz2>

MD5 sum: 9d22ee4daf3a194457caf4706f9cf01

- **Bison (2.3) - 1,060 KB:**

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

Download: <http://ftp.gnu.org/gnu/bison/bison-2.3.tar.bz2>

MD5 sum: c18640c6ec31a169d351e3117ecce3ec

- **Bzip2 (1.0.4) - 828 KB:**

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

Download: <http://www.bzip.org/1.0.4/bzip2-1.0.4.tar.gz>

MD5 sum: fc310b254f6ba5fbb5da018f04533688

- **CLFS-Bootscripts (1.1.0) - 40 KB:**

Download: <http://cross-lfs.org/files/packages/1.1.0/bootscripts-cross-lfs-1.1.0.tar.bz2>

MD5 sum: 1e6fcdfcc25cee2b53091a2d9f07e464

- **Coreutils (6.9) - 5,272 KB:**

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

Download: <http://ftp.gnu.org/gnu/coreutils/coreutils-6.9.tar.bz2>

MD5 sum: c9607d8495f16e98906e7ed2d9751a06

- **DejaGNU (1.4.4) - 1,056 KB:**

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

Download: <http://ftp.gnu.org/gnu/dejagnu/dejagnu-1.4.4.tar.gz>

MD5 sum: 053f18fd5d00873de365413cab17a666

- **Diffutils (2.8.7) - 1,034 KB:**

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

Download: <ftp://alpha.gnu.org/gnu/diffutils/diffutils-2.8.7.tar.gz>

MD5 sum: 18d6587cb915e7aa110a5d463d6ed156

- **E2fsprogs (1.40.4) - 3,960 KB:**

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

Download: <http://downloads.sourceforge.net/e2fsprogs/e2fsprogs-1.40.4.tar.gz>

MD5 sum: 124d744bdf9d443591eb8193c085944b

- **Expect (5.43.0) - 514 KB:**

Home page: <http://expect.nist.gov>

Download: <http://expect.nist.gov/src/expect-5.43.0.tar.gz>

MD5 sum: 43e1dc0e0bc9492cf2e1a6f59f276bc3

- **File (4.23) - 556 KB:**

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

Download: <http://cross-lfs.org/files/packages/1.1.0/file-4.23.tar.gz>

MD5 sum: 014a69979a8d5225a6ca2bcc4d7e967e



Note

File (4.23) 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/1.1.0/>.

• **Findutils (4.2.32) - 1,404 KB:**

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

Download: <http://ftp.gnu.org/gnu/findutils/findutils-4.2.32.tar.gz>

MD5 sum: aaa6beeb41a6f04963dff58f24a55b96

• **Flex (2.5.35) - 1232 KB:**

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

Download: <http://downloads.sourceforge.net/flex/flex-2.5.35.tar.bz2>

MD5 sum: 10714e50cea54dc7a227e3eddc44d57

• **Gawk (3.1.6) - 1,824 KB:**

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

Download: <http://ftp.gnu.org/gnu/gawk/gawk-3.1.6.tar.bz2>

MD5 sum: c9926c0bc8c177cb9579708ce67f0d75

• **GCC (4.2.4) - 43,044 KB:**

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

Download: <http://ftp.gnu.org/gnu/gcc/gcc-4.2.4/gcc-4.2.4.tar.bz2>

MD5 sum: d79f553e7916ea21c556329eacfeaa16

• **Gettext (0.17) - 11,388 KB:**

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

Download: <http://ftp.gnu.org/gnu/gettext/gettext-0.17.tar.gz>

MD5 sum: 58a2bc6d39c0ba57823034d55d65d606

• **Glibc (2.7) - 15624 KB:**

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

Download: <http://ftp.gnu.org/gnu/glibc/glibc-2.7.tar.bz2>

MD5 sum: 065c5952b439deba40083ccd67bcc8f7

• **Grep (2.5.3) - 608 KB:**

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

Download: <http://ftp.gnu.org/gnu/grep/grep-2.5.3.tar.bz2>

MD5 sum: 27061ce1fde82876970b6549a156da8b

• **Groff (1.19.2) - 2,836 KB:**

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

Download: <http://ftp.gnu.org/gnu/groff/groff-1.19.2.tar.gz>

MD5 sum: f7c9cf2e4b9967d3af167d7c9fadaae4

• **Gzip (1.3.12) - 456 KB:**

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

Download: <http://ftp.gnu.org/gnu/gzip/gzip-1.3.12.tar.gz>

MD5 sum: b5bac2d21840ae077e0217bc5e4845b1

• **Iana-Etc (2.20) - 192 KB:**

Home page: <http://www.sethworklein.net/projects/iana-etc>

Download: <http://www.sethworklein.net/projects/iana-etc/downloads/iana-etc-2.20.tar.bz2>

MD5 sum: 51d584b7b6115528c21e8ea32250f2b1

• **Inetutils (1.5) - 1360 KB:**

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

Download: <http://ftp.gnu.org/gnu/inetutils/inetutils-1.5.tar.gz>

MD5 sum: aeacd11d19bf25c89d4eff38346bdfb9

• **IPRoute2 (2.6.23) - 340 KB:**

Home page: <http://linux-net.osdl.org/index.php/Iproute2>

Download: <http://developer.osdl.org/dev/iproute2/download/iproute2-2.6.23.tar.bz2>

MD5 sum: 2e59da739ef19990408cf0a5cb0cae3e

• **Kbd (1.13) - 625 KB:**

Download: <http://ftp.altlinux.com/pub/people/legion/kbd/kbd-1.13.tar.bz2>

MD5 sum: 2639fdfe2412b29cbf6936d7db91cee3

• **Less (418) - 300 KB:**

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

Download: <http://www.greenwoodsoftware.com/less/less-418.tar.gz>

MD5 sum: b5864d76c54ddf4627fd57ab333c88b4

• **Libtool (1.5.26) - 2,900 KB:**

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

Download: <http://ftp.gnu.org/gnu/libtool/libtool-1.5.26.tar.gz>

MD5 sum: aa9c5107f3ec9ef4200eb6556f3b3c29

• **Linux (2.6.24.7) - 45,638 KB:**

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

Download: <http://www.kernel.org/pub/linux/kernel/v2.6/linux-2.6.24.7.tar.bz2>

MD5 sum: 40a73780d51525d28d36dec852c680c4

• **M4 (1.4.10) - 728 KB:**

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

Download: <http://ftp.gnu.org/gnu/m4/m4-1.4.10.tar.bz2>

MD5 sum: 0a35bab2f5d605e08083d7e3cbd4b8b0

• **Make (3.81) - 1,125 KB:**

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

Download: <http://ftp.gnu.org/gnu/make/make-3.81.tar.bz2>

MD5 sum: 354853e0b2da90c527e35aabb8d6f1e6

• **Man (1.6e) - 248 KB:**

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

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

MD5 sum: d8187cd756398baefc48ba7d60ff6a8a

• **Man-pages (3.01) - 904 KB:**

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

Download: <http://www.kernel.org/pub/linux/docs/manpages/Archive/man-pages-3.01.tar.bz2>

MD5 sum: ef311ffb0c4f43bad822345fb88498f7

• **Mktemp (1.5) - 69 KB:**

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

Download: <ftp://ftp.mktemp.org/pub/mktemp/mktemp-1.5.tar.gz>

MD5 sum: 9a35c59502a228c6ce2be025fc6e3ff2

• **Module-Init-Tools (3.4) - 192 KB:**

Home page: <http://www.kerneltools.org/KernelTools.org>

Download: <http://www.kernel.org/pub/linux/utils/kernel/module-init-tools/module-init-tools-3.4.tar.bz2>

MD5 sum: db6ac059e80e8dd4389dbe81ee61f3c6

• **Ncurses (5.6) - 2,352 KB:**

Home page: <http://dickey.his.com/ncurses>

Download: <ftp://invisible-island.net/ncurses/ncurses-5.6.tar.gz>

MD5 sum: b6593abe1089d6aab1551c105c9300e3

• **Patch (2.5.9) - 198 KB:**

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

Download: <ftp://alpha.gnu.org/gnu/diffutils/patch-2.5.9.tar.gz>

MD5 sum: dacfb618082f8d3a2194601193cf8716

• **Perl (5.8.8) - 9,887 KB:**

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

Download: <http://ftp.funet.fi/pub/CPAN/src/perl-5.8.8.tar.bz2>

MD5 sum: a377c0c67ab43fd96eeec29ce19e8382

• **Procps (3.2.7) - 280 KB:**

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

Download: <http://procps.sourceforge.net/procps-3.2.7.tar.gz>

MD5 sum: f490bca772b16472962c7b9f23b1e97d

• **Psmisc (22.6) - 284 KB:**

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

Download: <http://downloads.sourceforge.net/psmisc/psmisc-22.6.tar.gz>

MD5 sum: 2e81938855cf5cc38856bd4a31d79a4c

• **Readline (5.2) - 1,996 KB:**

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

Download: <http://ftp.gnu.org/gnu/readline/readline-5.2.tar.gz>

MD5 sum: e39331f32ad14009b9ff49cc10c5e751

• **Sed (4.1.5) - 781 KB:**

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

Download: <http://ftp.gnu.org/gnu/sed/sed-4.1.5.tar.gz>

MD5 sum: 7a1cbbbb3341287308e140bd4834c3ba

• **Shadow (4.1.2) - 1,660 KB:**

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

Download: <ftp://pkg-shadow.alioth.debian.org/pub/pkg-shadow/shadow-4.1.2.tar.bz2>

MD5 sum: ce90cbe9cba7f6673cb10cad49083c1c

• **Sysklogd (1.5) - 86 KB:**

Home page: <http://www.infodrom.org/projects/sysklogd>

Download: <http://www.infodrom.org/projects/sysklogd/download/sysklogd-1.5.tar.gz>

MD5 sum: e053094e8103165f98ddafe828f6ae4b

• **Sysvinit (2.86) - 97 KB:**

Download: <ftp://ftp.cistron.nl/pub/people/miquels/sysvinit/sysvinit-2.86.tar.gz>

MD5 sum: 7d5d61c026122ab791ac04c8a84db967

• **Tar (1.20) - 1,872 KB:**

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

Download: <http://ftp.gnu.org/gnu/tar/tar-1.20.tar.bz2>

MD5 sum: 1a7e17f27abf583b3b0bc059a827e68b

• **Tcl (8.4.16) - 3,560 KB:**

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

Download: <http://downloads.sourceforge.net/tcl/tcl8.4.16-src.tar.gz>

MD5 sum: f1c783a49547778d160e10be2fb5d76f

• **Texinfo (4.11) - 1628 KB:**

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

Download: <http://ftp.gnu.org/gnu/texinfo/texinfo-4.11.tar.bz2>

MD5 sum: c6bf13df4fbef8ce874aacd6a51e814

• **Tree (1.5.1.1) - 32 KB:**

Home page: <http://mama.indstate.edu/users/ice/tree>

Download: <ftp://mama.indstate.edu/linux/tree/old/tree-1.5.1.1.tgz>

MD5 sum: f8fc70a9260b50bf8b4613e9d5f5e589

• **Udev (124) - 204 KB:**

Home page: <http://www.kernel.org/pub/linux/utils/kernel/hotplug/udev.html>

Download: <http://www.kernel.org/pub/linux/utils/kernel/hotplug/udev-124.tar.bz2>

MD5 sum: 4da0471c0ca3a2a2a77692f67120c03d

• **Util-linux-ng (2.14) - 2,804 KB:**

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

Download: <http://www.kernel.org/pub/linux/utils/util-linux-ng/v2.14/util-linux-ng-2.14.tar.bz2>

MD5 sum: 23f227da49df36f33fe47e917e332cd8

• **Vim (7.1) - 6,728 KB:**

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

Download: <ftp://ftp.vim.org/pub/vim/unix/vim-7.1.tar.bz2>

MD5 sum: 44c6b4914f38d6f9aa959640b89da329

• **Vim (7.1) language files (optional) - 1,168 KB:**

Download: <ftp://ftp.vim.org/pub/vim/extra/vim-7.1-lang.tar.gz>

MD5 sum: 144aa049ba70621acf4247f0459f3ee7

• **Zlib (1.2.3) - 485 KB:**

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

Download: <http://www.zlib.net/zlib-1.2.3.tar.gz>

MD5 sum: debc62758716a169df9f62e6ab2bc634

Total size of these packages: about 200 MB

3.3. Additional Packages for x86_64

• **Bin86 (0.16.17) - 149 KB:**

Home page: <http://homepage.ntlworld.com/robert.debath>

Download: <http://homepage.ntlworld.com/robert.debath/dev86/bin86-0.16.17.tar.gz>

MD5 sum: c9e8d72dd2e7457b52d0e3164fc199a1

- **Lilo (22.8) - 431 KB:**

Home page: <http://lilo.go.dyndns.org>

Download: <http://home.san.rr.com/johninsd/pub/linux/lilo/lilo-22.8.src.tar.gz>

MD5 sum: 72765f2aafd20e23ecf07ebd22baeec7

Total size of these packages: about 580 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 Fixes - 416 KB:**

Download: <http://patches.cross-lfs.org/1.1.0/bash-3.2-fixes-8.patch>

MD5 sum: 9f57c9bd5d7278529a9b13d8fc36a02e

- **Binutils Branch Update Patch - 432 KB:**

Download: http://patches.cross-lfs.org/1.1.0/binutils-2.18-branch_update-3.patch

MD5 sum: afc0cc91efe7d2e5a14008d0625462ef

- **Binutils Posix Patch - 4 KB:**

Download: <http://patches.cross-lfs.org/1.1.0/binutils-2.18-posix-1.patch>

MD5 sum: f2020cb369c2facbfd8987fa2263f31f

- **Coreutils Suppress Uptime, Kill, Su Patch - 12 KB:**

Download: http://patches.cross-lfs.org/1.1.0/coreutils-6.9-suppress_uptime_kill_su-1.patch

MD5 sum: 8bc938ec7210ffff3885b37b1339dca45

- **Coreutils Uname Patch - 8 KB:**

Download: <http://patches.cross-lfs.org/1.1.0/coreutils-6.9-uname-1.patch>

MD5 sum: 865d271bec9039f45a3be0ada21edae0

- **Expect Spawn Patch - 6.9 KB:**

Download: <http://patches.cross-lfs.org/1.1.0/expect-5.43.0-spawn-2.patch>

MD5 sum: 7706e1e8238c72eed8dc905d6f3b6aa9

- **GCC Cross Search Paths Patch - 4 KB:**

Download: http://patches.cross-lfs.org/1.1.0/gcc-4.2.4-cross_search_paths-1.patch

MD5 sum: 2d316305d73bb14b3acd6a1f3a2ee4e9

- **GCC Posix Patch - 16 KB:**

Download: <http://patches.cross-lfs.org/1.1.0/gcc-4.2.4-posix-1.patch>

MD5 sum: 8df4fe6a9f77135ae1b0851786c40410

- **GCC PR31490 Patch - 1.4 KB:**

Download: <http://patches.cross-lfs.org/1.1.0/gcc-4.2.4-PR31490-1.patch>

MD5 sum: 43f3d086b57f02cb605819452936c4d7

- **Glibc Branch Update Patch - 64 KB:**

Download: http://patches.cross-lfs.org/1.1.0/glibc-2.7-branch_update-1A.patch

MD5 sum: dd1119b68f26308446d33ccde05732b4

• **Glibc Disable linking with libgcc_eh.a - 8 KB:**

Download: http://patches.cross-lfs.org/1.1.0/glibc-2.7-libgcc_eh-1.patch

MD5 sum: f868605171066a0d8f572314f6ea415c

• **Glibc Localedef Segfault - 4 KB:**

Download: http://patches.cross-lfs.org/1.1.0/glibc-2.7-localedef_segfault-1.patch

MD5 sum: 3ce6946b16e90a73d90ac701a5940709

• **Inetutils No-Server-Man-Pages Patch - 8 KB:**

Download: http://patches.cross-lfs.org/1.1.0/inetutils-1.5-no_server_man_pages-2.patch

MD5 sum: b04f46e66c4b438f609565073590ffd5

• **Linux Security Fixes Patch - 15.5 KB:**

Download: http://patches.cross-lfs.org/1.1.0/linux-2.6.24.7-security_fixes-1.patch

MD5 sum: 7f244856cc40e478f5f66a6a6267d4c6

• **Mktemp Tempfile Patch - 3.6 KB:**

Download: http://patches.cross-lfs.org/1.1.0/mktemp-1.5-add_tempfile-3.patch

MD5 sum: 65d73faabe3f637ad79853b460d30a19

• **Module-Init-Tools Manpages Patch - 35 KB:**

Download: <http://patches.cross-lfs.org/1.1.0/module-init-tools-3.4-manpages-1.patch>

MD5 sum: 2271047586981ae23adf01cc13d97791

• **Ncurses Bash Patch - 4 KB:**

Download: http://patches.cross-lfs.org/1.1.0/ncurses-5.6-bash_fix-1.patch

MD5 sum: c6f7f2ab0ebaf7721eb266641352db

• **Ncurses Branch Update Patch - 7188 KB:**

Download: http://patches.cross-lfs.org/1.1.0/ncurses-5.6-branch_update-3.patch

MD5 sum: 5c5436409af4699e8b81e4100b5203ec

• **Perl GCC 4.2.x Fix Patch - 4 KB:**

Download: http://patches.cross-lfs.org/1.1.0/perl-5.8.8-gcc42_fix-1.patch

MD5 sum: cd84d28231ea7f23d7e619d9304cc741

• **Perl Libc Patch - 4 KB:**

Download: <http://patches.cross-lfs.org/1.1.0/perl-5.8.8-libc-2.patch>

MD5 sum: 3bf8aef1fb6eb6110405e699e4141f99

• **Perl No page.h Patch - 4 KB:**

Download: http://patches.cross-lfs.org/1.1.0/perl-5.8.8-no_page_h-1.patch

MD5 sum: fe49a8a539b14f53adfe24482c1fbee1f

• **Perl Security Fixes Patch - 10 KB:**

Download: http://patches.cross-lfs.org/1.1.0/perl-5.8.8-security_fixes-1.patch

MD5 sum: 918a8f41d9e4fe1e272381e58be71497

• **Readline Fixes - 20 KB:**

Download: <http://patches.cross-lfs.org/1.1.0/readline-5.2-fixes-4.patch>

MD5 sum: 14ae1c3f3ff7223656f874bcf36e6f19

• **Texinfo mbstate Fix Patch - 4 KB:**

Download: http://patches.cross-lfs.org/1.1.0/texinfo-4.11-mbstate_fix-1.patch

MD5 sum: 37e3d170d809bb1c90e9741864839f66

- **Vim Branch Update Patch - 284 KB:**

Download: http://patches.cross-lfs.org/1.1.0/vim-7.1-branch_update-2.patch

MD5 sum: bd5ee693b6ed59567dc0e7c003d22694

- **Zlib fPIC Patch - 3.2 KB:**

Download: <http://patches.cross-lfs.org/1.1.0/zlib-1.2.3-fPIC-1.patch>

MD5 sum: 545d60b20bfde6f53023de44438cef59

Total size of these patches: about 8 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/1.1.0/> and acquire any additional patches to suit the system needs.

3.5. Additional Patches for x86_64

- **Bin86 x86_64 Patch - 2 KB:**

Download: http://patches.cross-lfs.org/1.1.0/bin86-0.16.17-x86_64-1.patch

MD5 sum: 92bdce7b0655cd2e9f83c83fc56d128e

- **GCC Pure 64 Patch - 8 KB:**

Download: <http://patches.cross-lfs.org/1.1.0/gcc-4.2.4-pure64-1.patch>

MD5 sum: 04d9b9cd2c3dad221d53a83d3a45d66f

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

Download: http://patches.cross-lfs.org/1.1.0/gcc-4.2.4-pure64_specs-1.patch

MD5 sum: a543edb70d0fbbebb4f365bdad0a9a393

- **Lilo x86_64 Cross Compile Patch - 4 KB:**

Download: http://patches.cross-lfs.org/1.1.0/lilo-22.8-cross_compile_x86_64-2.patch

MD5 sum: a7440e18ef7fa6c5d1c6181dc0cc4d10

Total size of these patches: about 34 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`.

`-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**.

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 - c1fs` 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 as user `c1fs`, and a `cd` into the created directory should be performed. The build instructions assume that the `bash` shell is in use.

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 Flags

We will need to setup target specific flags for the compiler and linkers.

```
export BUILD64="-m64"
```

Lets add the build flags to `~/ .bashrc` to prevent issues if we stop and come back later.

```
echo export BUILD64="\${BUILD64}" >> ~/ .bashrc
```

5.4. 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 $MACHINE | \
sed "s/$(echo $MACHINE | cut -d- -f2)/cross/")"
```

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

```
export CLFS_TARGET="x86_64-unknown-linux-gnu"
```

Copy settings to Environment

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

```
echo export CLFS_HOST="\${CLFS_HOST}" >> ~/ .bashrc
echo export CLFS_TARGET="\${CLFS_TARGET}" >> ~/ .bashrc
```

5.5. Linux-Headers-2.6.24.7

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 tarballs.

Install the header files that are common to all architectures:

```
install -dv /tools/include
make mrproper
make ARCH=x86_64 headers_check
make ARCH=x86_64 INSTALL_HDR_PATH=dest headers_install
cp -rv dest/include/* /tools/include
```

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

5.6. File-4.23

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
```

Compile the package:

```
make
```

Install the package:

```
make install
```

Details on this package are located in Section 10.30.2, “Contents of File.”

5.7. Cross Binutils-2.18

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

5.7.1. 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.

To make sure that the proper syntax is used for a couple of tools, apply the following patch:

```
patch -Np1 -i ../binutils-2.18-posix-1.patch
```

The following patches merges all updates from the 2.18 Branch from the Binutils developers:

```
patch -Np1 -i ../binutils-2.18-branch_update-3.patch
```

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

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

Prepare Binutils for compilation:

```
AR=ar AS=as ../binutils-2.18/configure --prefix=/cross-tools \
  --host=${CLFS_HOST} --target=${CLFS_TARGET} --with-lib-path=/tools/lib \
  --disable-nls --enable-shared --enable-64-bit-bfd --disable-multilib
```

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.

--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.

--enable-shared

Enable the creation of the shared libraries.

--disable-multilib

This option disables the building of a multilib capable Binutils.

```
--enable-64-bit-bfd
```

This adds 64 bit support to 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-2.18/include/libiberty.h /tools/include
```

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

5.8. Cross GCC-4.2.4 - Static

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

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

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

```
patch -Np1 -i ../gcc-4.2.4-pure64_specs-1.patch
```

To make sure that a couple of tools use the proper syntax, apply the following patch:

```
patch -Np1 -i ../gcc-4.2.4-posix-1.patch
```

The following patch ensures that `gcc` does not search the `/usr` directory for `libgcc_s.so` when cross-compiling:

```
patch -Np1 -i ../gcc-4.2.4-cross_search_paths-1.patch
```

Change the StartFile Spec to point to the correct library location:

```
echo "
#undef STARTFILE_PREFIX_SPEC
#define STARTFILE_PREFIX_SPEC \"/tools/lib/" >> gcc/config/linux.h
```

Now alter `gcc`'s `c` preprocessor's default include search path to use `/tools` only:

```
cp -v gcc/Makefile.in{,.orig}
sed -e "s@\(^CROSS_SYSTEM_HEADER_DIR =\) .*@ \1 /tools/include@g" \
    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
```

Prepare GCC for compilation:

```
../gcc-4.2.4/configure --prefix=/cross-tools \
  --host=${CLFS_HOST} --target=${CLFS_TARGET} --disable-multilib \
  --with-local-prefix=/tools --disable-nls --disable-shared \
  --disable-threads --enable-languages=c
```

The meaning of the configure options:

`--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.

`--disable-shared`

Disables the creation of the shared 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.

--enable-languages=c

This option ensures that only the C compiler is built.

Continue with compiling the package:

```
make all-gcc
```

Install the package:

```
make install-gcc
```

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

5.9. Glibc-2.7

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

5.9.1. Installation of Glibc

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

Disable linking to `libgcc_eh`:

```
patch -Np1 -i ../glibc-2.7-libgcc_eh-1.patch
```

The following patch fixes an issue that can cause `localdef` to segfault:

```
patch -Np1 -i ../glibc-2.7-localedef_segfault-1.patch
```

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

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

The following lines need to be added to `config.cache` for Glibc to support NPTL:

```
echo "libc_cv_forced_unwind=yes" > config.cache
echo "libc_cv_c_cleanup=yes" >> config.cache
```

Prepare Glibc for compilation:

```
BUILD_CC="gcc" CC="{CLFS_TARGET}-gcc {BUILD64}" \
  AR="{CLFS_TARGET}-ar" RANLIB="{CLFS_TARGET}-ranlib" \
  ../glibc-2.7/configure --prefix=/tools \
  --host={CLFS_TARGET} --build={CLFS_HOST} \
  --disable-profile --enable-add-ons \
  --with-tls --enable-kernel=2.6.0 --with-__thread \
  --with-binutils=/cross-tools/bin --with-headers=/tools/include \
  --cache-file=config.cache
```

The meaning of the new configure options:

```
BUILD_CC="gcc"
```

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

```
CC="{CLFS_TARGET}-gcc {BUILD64}"
```

Forces Glibc to build using our target architecture GCC utilizing the 64 Bit flags.

```
AR="{CLFS_TARGET}-ar"
```

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

```
RANLIB="{CLFS_TARGET}-ranlib"
```

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

`--disable-profile`

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

`--enable-add-ons`

This tells Glibc to utilize all add-ons that are available.

`--with-tls`

This tells Glibc to use Thread Local Storage.

`--enable-kernel=2.6.0`

This tells Glibc to compile the library with support for 2.6.x Linux kernels.

`--with-__thread`

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

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

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

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 Glibc.”

5.10. Cross GCC-4.2.4 - Final

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

5.10.1. Installation of GCC Cross Compiler

The following patch enables fixes compilation errors with the kernel:

```
patch -Np1 -i ../gcc-4.2.4-PR31490-1.patch
```

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

```
patch -Np1 -i ../gcc-4.2.4-pure64_specs-1.patch
```

To make sure that a couple of tools use the proper syntax, apply the following patch:

```
patch -Np1 -i ../gcc-4.2.4-posix-1.patch
```

The following patch ensures that `gcc` does not search the `/usr` directory for `libgcc_s.so` when cross-compiling:

```
patch -Np1 -i ../gcc-4.2.4-cross_search_paths-1.patch
```

Change the StartFile Spec to point to the correct library location:

```
echo "  
#undef STARTFILE_PREFIX_SPEC  
#define STARTFILE_PREFIX_SPEC \"/tools/lib/" >> gcc/config/linux.h
```

Now alter `gcc`'s c preprocessor's default include search path to use `/tools` only:

```
cp -v gcc/Makefile.in{,.orig}  
sed -e "s@\(^\^CROSS_SYSTEM_HEADER_DIR =\) .*@\1 /tools/include@g" \  
    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
```

Prepare GCC for compilation:

```
../gcc-4.2.4/configure --prefix=/cross-tools \  
  --target=${CLFS_TARGET} --host=${CLFS_HOST} --disable-multilib \  
  --with-local-prefix=/tools --disable-nls --enable-shared \  
  --enable-languages=c,c++ --enable-__cxa_atexit \  
  --enable-c99 --enable-long-long --enable-threads=posix
```

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 10.10.2, “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. Binutils-2.18

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

6.3.1. Installation of Binutils

To make sure that the proper syntax is used for a couple of tools, apply the following patch:

```
patch -Np1 -i ../binutils-2.18-posix-1.patch
```

The following patches merges all updates from the 2.18 Branch from the Binutils developers:

```
patch -Np1 -i ../binutils-2.18-branch_update-3.patch
```

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

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

Prepare Binutils for compilation:

```
CC="${CC} ${BUILD64}" \
  ../binutils-2.18/configure --prefix=/tools \
  --build=${CLFS_HOST} --host=${CLFS_TARGET} --target=${CLFS_TARGET} \
  --disable-nls --enable-shared --enable-64-bit-bfd \
  --disable-multilib
```

The meaning of the new configure options:

```
CC="${CC} ${BUILD64}"
```

Tells the compiler to use our 64-bit build flags.

Compile the package:

```
make configure-host
make
```

Install the package:

```
make install
```

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

6.4. GCC-4.2.4

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

6.4.1. Installation of GCC

The following patch enables fixes compilation errors with the kernel:

```
patch -Np1 -i ../gcc-4.2.4-PR31490-1.patch
```

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

```
patch -Np1 -i ../gcc-4.2.4-pure64_specs-1.patch
```

To make sure that a couple of tools use the proper syntax, apply the following patch:

```
patch -Np1 -i ../gcc-4.2.4-posix-1.patch
```

The following patch ensures that `gcc` does not search the `/usr` directory for `libgcc_s.so` when cross-compiling:

```
patch -Np1 -i ../gcc-4.2.4-cross_search_paths-1.patch
```

Now we will change `cpp`'s search path so it won't look in `/usr/include`:

```
cp -v gcc/cppdefault.c{,.orig}
sed -e '/#define STANDARD_INCLUDE_DIR/s@"/usr/include"@0@g' \
    gcc/cppdefault.c.orig > gcc/cppdefault.c
```

Also, we need to set the directory searched by the `fixincludes` process for system headers, so it won't look at the host's headers:

```
cp -v gcc/Makefile.in{,.orig}
sed -e 's@(^NATIVE_SYSTEM_HEADER_DIR =\).*@1 /tools/include@g' \
    gcc/Makefile.in.orig > gcc/Makefile.in
```

When searching for the multilibs, force the build to use the results of `--print-multi-lib` from our cross-compiler, not the native compiler `gcc` builds now:

```
cp -v gcc/Makefile.in{,.orig2}
sed -e "/MULTILIBS/s@\$(GCC_FOR_TARGET)@/cross-tools/bin/\${CC}@g" \
    gcc/Makefile.in.orig2 > 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:

```
CC="${CC} ${BUILD64}" CXX="${CXX} ${BUILD64}" \
  ../gcc-4.2.4/configure --prefix=/tools --disable-multilib \
  --build=${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-nls \
  --enable-__cxa_atexit --enable-languages=c,c++ --disable-libstdcxx-pch
```

The meaning of the new configure options:

```
CXX="${CXX} ${BUILD64}"
```

This forces the C++ compiler to use our 64 Bit flags.

```
--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.

Compile the package:

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

Install the package:

```
make install
```

Now we copy the files that are placed in `/tools/lib64` to `/tools/lib`. We also delete the `/tools/lib64` directory:

```
cp -va /tools/lib64/* /tools/lib
rm -rvf /tools/lib64
```

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

6.5. Ncurses-5.6

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

6.5.1. Installation of Ncurses

The following patch fixes an issue with some Bash versions:

```
patch -Np1 -i ../ncurses-5.6-bash_fix-1.patch
```

Prepare Ncurses for compilation:

```
CC="${CC} ${BUILD64}" CXX="${CXX} ${BUILD64}" \
./configure --prefix=/tools --with-shared --build=${CLFS_HOST} \
--host=${CLFS_TARGET} --without-debug --without-ada \
--enable-overwrite --with-build-cc=gcc
```

The meaning of the 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 10.17.2, “Contents of Ncurses.”

6.6. Bash-3.2

The Bash package contains the Bourne-Again SHell.

6.6.1. 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-3.2-fixes-8.patch
```

When bash is cross-compiled, it cannot test for the presence of named pipes, among other things. If you used **su** to become an unprivileged user, this combination will cause Bash to build without *process substitution*, which will break one of the c++ test scripts in `glibc`. The following prevents future problems by skipping 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:

```
CC="${CC} ${BUILD64}" CXX="${CXX} ${BUILD64}" \
./configure --prefix=/tools \
--build=${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 10.27.2, “Contents of Bash.”

6.7. Bzip2-1.0.4

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**.

6.7.1. 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}" ${BUILD64}" AR="${AR}" RANLIB="${RANLIB}"
```

Install the package:

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

Details on this package are located in Section 10.28.2, “Contents of Bzip2.”

6.8. Coreutils-6.9

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

6.8.1. Installation of Coreutils

The following command renames the internal implementation of futimens to gl_futimens as newer versions of Glibc provide an incompatible version:

```
sed -i "s/futimens/gl_&/" $(grep -lr futimens *)
```

Configure can not properly determine how to get free space when cross-compiling, Without putting the following entry into `config.cache` the `df` program will not be built.

```
echo "fu_cv_sys_stat_statfs2_bsize=yes" > config.cache
```

Prepare Coreutils for compilation:

```
CC="${CC} ${BUILD64}" ./configure --prefix=/tools \
  --build=${CLFS_HOST} --host=${CLFS_TARGET} \
  --cache-file=config.cache
```

When we changed the futimens references, one of the files we updated was `touch.c`. That change means the `man` Makefile will try to regenerate the man page for `touch`, by executing the compiled `touch` program. If the target architecture cannot be executed by the host system, the build will fail. The following command fixes this by giving the man page a newer timestamp than its corresponding source file:

```
touch man/touch.1
```

Compile the package:

```
make
```

Install the package:

```
make install
```

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

6.9. Diffutils-2.8.7

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

6.9.1. Installation of Diffutils

Prepare Diffutils for compilation:

```
CC="${CC} ${BUILD64}" ./configure --prefix=/tools \  
  --build=${CLFS_HOST} --host=${CLFS_TARGET}
```

Compile the package:

```
make
```

Install the package:

```
make install
```

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

6.10. Findutils-4.2.32

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).

6.10.1. Installation of Findutils

Prepare Findutils for compilation:

```
CC="${CC} ${BUILD64}" ./configure --prefix=/tools \  
  --build=${CLFS_HOST} --host=${CLFS_TARGET}
```

Compile the package:

```
make
```

Install the package:

```
make install
```

Details on this package are located in Section 10.31.2, “Contents of Findutils.”

6.11. Zlib-1.2.3

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

6.11.1. Installation of Zlib

Prepare Zlib for compilation:

```
CC="${CC} ${BUILD64}" \  
./configure --prefix=/tools --shared
```

Compile the package:

```
make
```

Install the package:

```
make install
```

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

6.12. File-4.23

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

6.12.1. Installation of File

Prepare File for compilation:

```
CC="${CC} ${BUILD64}" ./configure --prefix=/tools \  
--build=${CLFS_HOST} --host=${CLFS_TARGET}
```

Compile the package:

```
make
```

Install the package:

```
make install
```

Details on this package are located in Section 10.30.2, “Contents of File.”

6.13. Gawk-3.1.6

The Gawk package contains programs for manipulating text files.

6.13.1. Installation of Gawk

Prepare Gawk for compilation:

```
CC="${CC} ${BUILD64}" \  
./configure --prefix=/tools \  
--build=${CLFS_HOST} --host=${CLFS_TARGET}
```

Compile the package:

```
make
```

Install the package:

```
make install
```

Details on this package are located in Section 10.32.2, “Contents of Gawk.”

6.14. Gettext-0.17

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.

6.14.1. 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:

```
CC="${CC} ${BUILD64}" CXX="${CXX} ${BUILD64}" \
./configure --prefix=/tools --disable-shared \
--build=${CLFS_HOST} --host=${CLFS_TARGET} \
--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 10.33.2, “Contents of Gettext.”

6.15. Grep-2.5.3

The Grep package contains programs for searching through files.

6.15.1. Installation of Grep

Prepare Grep for compilation:

```
CC="${CC} ${BUILD64}" ./configure --prefix=/tools \  
  --build=${CLFS_HOST} --host=${CLFS_TARGET} \  
  --disable-perl-regexp
```

The meaning of the configure options:

--disable-perl-regexp

This ensures that the **grep** program does not get linked against a Perl Compatible Regular Expression (PCRE) library that may be present on the host but will not be available when building the final system.

Compile the package:

```
make
```

Install the package:

```
make install
```

Details on this package are located in Section 10.34.2, “Contents of Grep.”

6.16. Gzip-1.3.12

The Gzip package contains programs for compressing and decompressing files.

6.16.1. Installation of Gzip

The following command renames the internal implementation of futimens to gl_futimens as newer versions of Glibc provide an incompatible version:

```
sed -i "s/futimens/gl_&/" $(grep -lr futimens *)
```

Prepare Gzip for compilation:

```
CC="${CC} ${BUILD64}" ./configure --prefix=/tools \  
--build=${CLFS_HOST} --host=${CLFS_TARGET}
```

Compile the package:

```
make
```

Install the package:

```
make install
```

Details on this package are located in Section 10.36.2, “Contents of Gzip.”

6.17. Make-3.81

The Make package contains a program for compiling packages.

6.17.1. Installation of Make

Prepare Make for compilation:

```
CC="${CC} ${BUILD64}" ./configure --prefix=/tools \  
--build=${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 Make.”

6.18. Patch-2.5.9

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

6.18.1. Installation of Patch

Prepare Patch for compilation:

```
CC="${CC} ${BUILD64}" ./configure --prefix=/tools \  
--build=${CLFS_HOST} --host=${CLFS_TARGET}
```

Compile the package:

```
make
```

Install the package:

```
make install
```

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

6.19. Sed-4.1.5

The Sed package contains a stream editor.

6.19.1. Installation of Sed

Prepare Sed for compilation:

```
CC="${CC} ${BUILD64}" ./configure --prefix=/tools \  
  --build=${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 Sed.”

6.20. Tar-1.20

The Tar package contains an archiving program.

6.20.1. Installation of Tar

TAR utilizes UTF-8 capability, without the following command the UTF-8 sections of the build will fail:

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

Prepare Tar for compilation:

```
CC="${CC} ${BUILD64}" ./configure --prefix=/tools \  
  --build=${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 10.49.2, “Contents of Tar.”

6.21. Texinfo-4.11

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

6.21.1. Installation of Texinfo

The following patch fixes a build error due to mbstate being undefined:

```
patch -Np1 -i ../texinfo-4.11-mbstate_fix-1.patch
```

Prepare Texinfo for compilation:

```
CC="${CC} ${BUILD64}" ./configure --prefix=/tools \  
  --build=${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 10.50.2, “Contents of Texinfo.”

6.22. 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.x (having been compiled with GCC-3.0 or greater). The reason for the kernel version requirement is that, without it, thread-local storage support in Binutils will not be built and the Native POSIX Threading Library (NPTL) test suite will segfault.

To check your kernel version, run **cat /proc/version** - if it does not say that you are running a 2.6.2 or later Linux kernel, compiled with GCC 3.0 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},sbin,src,sys}
mkdir -pv ${CLFS}/var/{lock,log,mail,run,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,pwd,rm,stty} ${CLFS}/bin
ln -sv /tools/bin/file ${CLFS}/usr/bin
ln -sv /tools/lib/libgcc_s.so{,.1} ${CLFS}/usr/lib
ln -sv /tools/lib/libstd*so* ${CLFS}/usr/lib
ln -sv bash ${CLFS}/bin/sh
```

To enable some c++ tests in the glibc and binutils test suites to link, create a directory and make some symbolic links:

```
mkdir -pv ${CLFS}/usr/lib64
ln -sv /tools/lib/libstd*so* ${CLFS}/usr/lib64
```

7.4. E2fsprogs-1.40.4

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

7.4.1. Installation of E2fsprogs

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

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

Prepare E2fsprogs for compilation:

```
CC="${CC} ${BUILD64}" ../configure --prefix=/tools \
  --enable-elf-shlibs --disable-evms --with-cc="${CC} ${BUILD64}" \
  --with-linker=${LD} --host=${CLFS_TARGET}
```

The meaning of the configure options:

--enable-elf-shlibs

This creates the shared libraries which some programs in this package use.

--disable-evms

This disables the building of the Enterprise Volume Management System (EVMS) plugin. This plugin is not up-to-date with the latest EVMS internal interfaces and EVMS is not installed as part of a base CLFS system, so the plugin is not required. See the EVMS website at <http://evms.sourceforge.net/> for more information regarding EVMS.

Compile the package:

```
make
```

Install the binaries, documentation and shared libraries:

```
make DESTDIR="${CLFS}" 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,e2fsck} ${CLFS}/sbin
```

Details on this package are located in Section 10.12.2, “Contents of E2fsprogs.”

7.5. Sysvinit-2.86

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

7.5.1. Installation of Sysvinit

Make some modifications to allow you to boot into the minimal temp-system:

```
cp -v src/Makefile src/Makefile.orig
sed -e 's@root@0@g' \
    -e "s@/dev/initctl@${CLFS}&@g" \
    -e 's@(mknod \)-m \([0-9]* \)\(.* \)p@1\3p; chmod \2\3@g' \
    -e "s@/usr/lib@/tools/lib@" \
    src/Makefile.orig > src/Makefile
```

Compile the package:

```
make -C src clobber
make -C src CC="${CC} ${BUILD64}"
```

Install the package:

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

7.5.2. 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

l0:0:wait:/etc/rc.d/init.d/rc 0
l1:S1:wait:/etc/rc.d/init.d/rc 1
l2:2:wait:/etc/rc.d/init.d/rc 2
l3:3:wait:/etc/rc.d/init.d/rc 3
l4:4:wait:/etc/rc.d/init.d/rc 4
l5:5:wait:/etc/rc.d/init.d/rc 5
l6:6:wait:/etc/rc.d/init.d/rc 6

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

su:S016:once:/sbin/sulogin

1:2345:respawn:/sbin/agetty -I '\033(K' tty1 9600
2:2345:respawn:/sbin/agetty -I '\033(K' tty2 9600
3:2345:respawn:/sbin/agetty -I '\033(K' tty3 9600
4:2345:respawn:/sbin/agetty -I '\033(K' tty4 9600
5:2345:respawn:/sbin/agetty -I '\033(K' tty5 9600
6:2345:respawn:/sbin/agetty -I '\033(K' tty6 9600

# End /etc/inittab
EOF
```

Details on this package are located in Section 10.48.3, “Contents of Sysvinit.”

7.6. Module-Init-Tools-3.4

The Module-Init-Tools package contains programs for handling kernel modules in Linux kernels greater than or equal to version 2.5.47.

7.6.1. Installation of Module-Init-Tools

The tarball only contains sgml source for the manual pages. The following patch contains the result of processing this through **docbook2man** which we do not build as part of a basic clfs install :

```
patch -Np1 -i ../module-init-tools-3.4-manpages-1.patch
```

Prepare Module-Init-Tools for compilation:

```
CC="${CC} ${BUILD64}" ./configure --prefix=/ \
  --build=${CLFS_HOST} --host=${CLFS_TARGET}
```

Compile the package:

```
make
```

Install the package:

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

Details on this package are located in Section 10.43.2, “Contents of Module-Init-Tools.”

7.7. Util-linux-ng-2.14

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

7.7.1. Installation of Util-linux-ng

Prepare Util-linux-ng for compilation:

```
CC="${CC} ${BUILD64}" ./configure --build=${CLFS_HOST} \  
  --host=${CLFS_TARGET} --enable-login-utils \  
  --disable-makeinstall-chown
```

Compile the package:

```
make
```

Install the package:

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

Details on this package are located in Section 10.52.3, “Contents of Util-linux-ng.”

7.8. Udev-124

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

7.8.1. Installation of Udev

Compile the package:

```
make CROSS_COMPILE="${CLFS_TARGET}-" CC="${CC} ${BUILD64}" \  
LD="${CC} ${BUILD64}"
```

Install the package:

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

Details on this package are located in Section 10.51.2, “Contents of Udev.”

7.9. 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 `${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 allow 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

```
uucp:x:32:32:uucp:/var/spool/uucp:/bin/false
```

Often used for Unix-to-Unix Copy of files from one server to the next

```
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:4:
tape:x:5:
daemon:x:6:
floppy:x:7:
disk:x:8:
lp:x:9:
dialout:x:10:
audio:x:11:
video:x:12:
utmp:x:13:
usb:x:14:
cdrom:x:15:
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

`uucp:x:32:uucp`

Used by the Unix-to-Unix copy users

`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 Udev 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.

7.10. Linux-2.6.24.7

The Linux package contains the Linux kernel.

7.10.1. 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.

A number of vulnerabilities have come to light after the stable kernel team stopped supporting 2.6.24. The following patch addresses them:

```
patch -Np1 -i ../linux-2.6.24.7-security_fixes-1.patch
```

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.

To ensure that your system boots and you can properly run both 32 bit and 64 bit binaries, please make sure that you enable support for ELF and emulations for 32bit ELF into 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=x86_64 CROSS_COMPILE=${CLFS_TARGET}- menuconfig
```

Compile the kernel image and modules:

```
make ARCH=x86_64 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 Udev are not documented. The problem is that Udev 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 Udev:

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

Install the modules, if the kernel configuration uses them:

```
make ARCH=x86_64 CROSS_COMPILE=${CLFS_TARGET}- \
INSTALL_MOD_PATH=${CLFS} modules_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 arch/x86_64/boot/bzImage ${CLFS}/boot/clfskernel-2.6.24.7
```

`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-2.6.24.7
```

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-2.6.24.7
```

Details on this package are located in Section 12.3.2, “Contents of Linux.”

7.11. Building a bootloader

On x86 and x86_64 (multilib) architectures, the preferred bootloader is GRUB. Unfortunately, GRUB doesn't work on x86_64 Pure64 - the stage2 files can be correctly built as 32-bit, but the **grub** shell is a 64-bit program, and tries to execute some of the stage2 routines - this results in a segmentation fault. Therefore, in the final system we use Lilo as the bootloader.

If you already have a bootloader, such as GRUB, installed on the system then you should use that to make your new kernel bootable.



Note

We will now cross-compile Bin86 and Lilo - these instructions assume you are using an x86_64 machine (e.g. booted from a Live CD), either pure64 or multilib. This approach will not work if you are running the machine as i686, because a 32-bit kernel will not be able to execute a 64-bit binary to install the bootloader. If that is the case, you will need to install an i686 bootloader on the host system.

7.12. Bin86-0.16.17

The Bin86 package contains an assembler and linker to produce 16 or 32-bit 8086 machine code.

7.12.1. Installation of Bin86

We are building Bin86 so that we can compile Lilo. Both **as86** and **ld86** need to run on the host system to assemble x86_64 code. We cannot compile the whole package like this, but fortunately these two programs are the only parts we require.

This patch updates Bin86 to compile on x86_64:

```
patch -Np1 -i ../bin86-0.16.17-x86_64-1.patch
```

The Bin86 package does not contain a **configure** script. Natively compile only the necessary parts with:

```
make CC=gcc -C as as86
make CC=gcc -C ld ld86
```

Install the assembler and linker where they will be on the PATH when we build Lilo, using prefixes to show that their output is not for a native system.

```
install -v -m 755 -s as/as86 /cross-tools/bin/${CLFS_TARGET}-as86
install -v -m 755 -s ld/ld86 /cross-tools/bin/${CLFS_TARGET}-ld86
```

Details on this package are located in Section 10.54.2, “Contents of Bin86.”

7.13. Lilo-22.8

The Lilo package contains the Linux Loader, a bootloader.

We have chosen to use Lilo because at the moment no other bootloader builds and runs on a pure64 system. If your machine has multiple systems (i.e. x86 or multilib) you may prefer to use the bootloader from the other system, such as GRUB.

7.13.1. Installation of Lilo

The following patch forces Lilo to use our cross-compiler (except for a couple of utilities which run during the compile), and to look for as86 and ld86 under the names by which we installed them.

```
patch -Np1 -i ../lilo-22.8-cross_compile_x86_64-2.patch
```

Compile the program which installs the loader, linking it statically so that it does not have to link to host libraries when it runs:

```
make lilo.static
```

Install only the parts we need:

```
install -v -m755 lilo.static /tools/bin  
install -v -m755 keytab-lilo.pl /tools/bin
```

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

7.14. 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` 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.

7.15. Build Flags

We will need to copy our build variables into our new system:

```
echo export BUILD64=\"\"${BUILD64}\"\" >> ${CLFS}/root/.bash_profile
```

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=4,mode=620 0     0
shm           /dev/shm     tmpfs  defaults  0     0
# End /etc/fstab
EOF
```

Replace `[xxx]`, `[yyy]`, and `[fff]` with the values appropriate for the system, for example, `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. CLFS-Bootscripts-1.1.0

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

7.17.1. Installation of CLFS-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 CLFS-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. Also, double-check that `${CLFS}` is set as `root`.

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:

```
mkknod -m 600 ${CLFS}/dev/console c 5 1
mkknod -m 666 ${CLFS}/dev/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 root:root ${CLFS}
```

7.20. Making the CLFS System Bootable

You are nearly ready to boot to the new temporary system. One of the last things to do is to ensure that the system can be booted. The instructions below apply only to `x86_64` machines on which Lilo is going to be installed. Information on using a pre-installed GRUB on machines currently running as `x86` or `x86_64` multilib should be available in the usual resource-specific locations for those architectures. If you have installed an `x86` version of Lilo, these instructions should be modified to refer to the `lilo` and `keytab-lilo.pl` commands you installed on the host system.

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

If you have multiple systems on your machine using a different bootloader such as GRUB, you may prefer to use that instead - consult the appropriate documentation. The rest of this section assumes you are going to use Lilo.

Earlier, we compiled and installed the Lilo boot loader software in preparation for this step. The procedure involves writing a boot image to a specific location on the hard drive. If you have a floppy disk drive, or if you have installed a cd recording program, we highly recommend using `mkrescue` to create a Lilo boot floppy, or CD (using e.g. `dvdrecord` from `dvdrttools`) as a test and as a backup.

Normally, you interact with Lilo by using the cursor and `enter` keys to select from the available option(s), but sometimes it is necessary to add other boot options, such as e.g. `'init=/bin/bash'` to debug boot failures. The more your keyboard layout differs from the US qwerty layout, the harder it becomes to type boot options unless Lilo knows about your keyboard layout. So, we will create a key table for Lilo (`.ctl`) file - at one point in the documentation these

are referred to as .klt files, which may be a typo, but has been followed by some distros. The name, and location, are not important but it is conventional to put these in /boot with the name representing the key layout. For a British keyboard layout, the following command will achieve this:

```
keytab-lilo.pl uk >${CLFS}/boot/uk.ktl
```

The argument to the command is the name of the keymap, or if necessary you can specify the full path to the keymap. Use whatever is appropriate for your keyboard.

The next step is to create /etc/lilo.conf:

```
cat > ${CLFS}/etc/lilo.conf << "EOF"
# Begin /etc/lilo.conf
# lilo.conf
#
# global options
boot=/dev/<bootdisk>
keytable=/boot/<keytable>
lba32
map=/boot/map
prompt

# set the name of the default image to boot
default=clfs

# define an image
image=/boot/clfskernel
    label=clfs
    root=/dev/<partition>
    read-only
# optionally add parameters to pass, e.g.
#   append="video=radeonfb:1024x768-16@70"

# if you had an existing system, you could
# add its details here.
EOF
```

Replace <bootdisk> with the name of the disk (or partition) on which the boot sector is to be written, e.g. sda. Replace <keytable> with the name of the keytable file you created, and <partition> with the name of the root partition for the new system.



Warning

The following command will overwrite any current boot loader. Do not run the command if this is not desired. If you have cross-compiled on a different machine from the target, you must install the bootloader on the target machine, the installed boot block is not a file which can be copied with **tar**.

Run Lilo:

```
/tools/bin/lilo.static -v -C ${CLFS}/etc/lilo.conf
```



Note

People who have been used to GRUB need to be aware that Lilo works differently - in particular, you cannot edit the available choices as you can in the **grub** shell, and Lilo records the block addresses of the kernels into the boot blocks each time `/sbin/lilo` is run. This means that when you compile a new kernel, you have to add it to `/etc/lilo.conf` and rerun `/sbin/lilo`. It also means that if you recompile an existing kernel and save it to the same name you still have to rerun `/sbin/lilo` in case it now occupies different blocks on the filesystem.

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. E2fsprogs-1.40.4

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

8.2.1. 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:

```
CC="${CC} ${BUILD64}" ../configure --prefix=/tools \
  --enable-elf-shlibs --disable-evms --with-cc="${CC} ${BUILD64}" \
  --with-linker=${LD} --host=${CLFS_TARGET}
```

The meaning of the configure options:

--enable-elf-shlibs

This creates the shared libraries which some programs in this package use.

--disable-evms

This disables the building of the Enterprise Volume Management System (EVMS) plugin. This plugin is not up-to-date with the latest EVMS internal interfaces and EVMS is not installed as part of a base CLFS system, so the plugin is not required. See the EVMS website at <http://evms.sourceforge.net/> for more information regarding EVMS.

Compile the package:

```
make libs
```

Install the static libraries and headers:

```
make install-libs
```

The meaning of the make option:

install-libs

This command will only install the e2fsprogs libraries.

Details on this package are located in Section 10.12.2, “Contents of E2fsprogs.”

8.3. Util-linux-ng-2.14

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

8.3.1. Installation of Util-linux-ng

Prepare Util-linux-ng for compilation:

```
CC="${CC} ${BUILD64}" ./configure --prefix=/tools \  
  --build=${CLFS_HOST} --host=${CLFS_TARGET}
```

Compile the package:

```
make -C mount mount umount  
make -C text-utils more
```

Install the package:

```
cp -v mount/{,u}mount text-utils/more /tools/bin
```

Details on this package are located in Section 10.52.3, “Contents of Util-linux-ng.”

8.4. Mounting Virtual Kernel File Systems



Note

The commands in the remainder of the book should be run as the `root` user. Also, double-check that `${CLFS}` is set as `root`.

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 Udev early in the boot process, so we create them here:

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

Once the system is complete and booting, the rest of our device nodes are created by the Udev 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:

```
mount -f -vt tmpfs tmpfs ${CLFS}/dev/shm
mount -f -vt devpts -o gid=4,mode=620 devpts ${CLFS}/dev/pts
```

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

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},sbin,srv,sys}
mkdir -pv /var/{lock,log,mail,run,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,rm,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
```

To enable some c++ tests in the glibc and binutils testsuites to link, create a directory and make some symbolic links:

```
mkdir -pv /usr/lib64
ln -sv /tools/lib/libstd*so* /usr/lib64
```

8.9. Build Flags

We will need to setup target specific flags for the compiler and linkers.

```
export BUILD64="-m64"
```

To prevent errors when you come back to your build, we will export these variables to prevent any build issues in the future:

```
echo export BUILD64=\"\"${BUILD64}\"\" >> ~/.bash_profile
```

8.10. 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 /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

```
uucp:x:32:32:uucp:/var/spool/uucp:/bin/false
```

Often used for Unix-to-Unix Copy of files from one server to the next

```
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:4:
tape:x:5:
daemon:x:6:
floppy:x:7:
disk:x:8:
lp:x:9:
dialout:x:10:
audio:x:11:
video:x:12:
utmp:x:13:
usb:x:14:
cdrom:x:15:
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

```
uucp:x:32:uucp
```

Used by the Unix-to-Unix copy users

```
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 Udev 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.

8.11. Mounting Kernel Filesystems

8.11.1. Mounting Additional Kernel Filesystems

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

```
mount -vt devpts -o gid=4,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 but is also not required for the file systems to be properly mounted. As such, 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 to run the tests that the packages have. I.e., **make check**. Tcl, Expect, and DejaGNU are needed for the GCC and Binutils test suites, and Tree is used by the Udev test suite. 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-8.4.16

The Tcl package contains the Tool Command Language.

9.2.1. Installation of Tcl

Prepare Tcl for compilation:

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

Build the package:

```
make
```

Install the package:

```
make install
```

Tcl's private header files are needed for the next package, Expect. Install them into /tools:

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

Now make a necessary symbolic link:

```
ln -sv tclsh8.4 /tools/bin/tclsh
```

9.2.2. Contents of Tcl

Installed programs: tclsh (link to tclsh8.4) and tclsh8.4

Installed library: libtcl8.4.so

Short Descriptions

tclsh8.4 The Tcl command shell

tclsh A link to tclsh8.4

libtcl8.4.so The Tcl library

9.3. Expect-5.43.0

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

9.3.1. Installation of Expect

The following sed tells **configure** to look for libraries in `${libdir}`, not just in `/tools/lib`:

```
sed -i '/EXP_LIB_SPEC=/s@${exec_prefix}/lib@${libdir}@"' configure
```

Fix a bug that can result in false failures during the GCC test suite run:

```
patch -Np1 -i ../expect-5.43.0-spawn-2.patch
```

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
<code>libexpect-5.43.a</code>	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-1.4.4

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. Tree-1.5.1.1

The Tree package contains a program that lists the directory structure in a graphical "tree" format. It is used by Udev for failures in its testsuite.

9.5.1. Installation of Tree

Compile the package:

```
make
```

Install the package:

```
make prefix=/tools install
```

The meaning of the make parameters:

prefix=/tools

This overrides the default prefix of /usr in the Makefile.

9.5.2. Contents of Tree

Installed program: tree

Short Descriptions

tree Tree will list contents of directories in a tree-like format.

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/optimization.txt>. 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 BLFS include:

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

There are some hints written on the topic of package management. Visit the *Hints 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 the CLFS and BLFS Book can be used to upgrade to the newer versions. Here are some points that you should be aware of when upgrading packages, especially on a running system.

- If one of the toolchain packages (Glibc, GCC or Binutils) needs to be upgraded to a newer minor version, it is safer to rebuild 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/fakeroot.txt>.

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/clfs-1.1.0/> to verify whether or not these failures are expected. This site is valid for all tests throughout this book.

10.4. Temporary Perl-5.8.8

The Perl package contains the Practical Extraction and Report Language.

10.4.1. Installation of Perl

The following patch causes `makedepend.sh` to also remove `<command-line>` as well as `<command line>`:

```
patch -Np1 -i ../perl-5.8.8-gcc42_fix-1.patch
```

The following patch prevents perl from including `asm/page.h` which will no longer be available in headers provided by kernel 2.6.25 and forward:

```
patch -Np1 -i ../perl-5.8.8-no_page_h-1.patch
```

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

```
patch -Np1 -i ../perl-5.8.8-libc-2.patch
```

Prepare Perl for compilation (make sure to get the 'Data/Dumper Fcntl IO POSIX' part of the command correct—they are all letters):

```
./configure.gnu --prefix=/tools \  
-Dstatic_ext='Data/Dumper IO Fcntl POSIX' -Dcc="gcc"
```

The meaning of the configure option:

```
-Dstatic_ext='Data/Dumper IO Fcntl POSIX'
```

This tells Perl to build the minimum set of static extensions needed for installing and testing the Glibc and Coreutils packages later in this chapter.

Now we are ready to make our perl utilities:

```
make perl utilities
```

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

Install these tools and their libraries:

```
cp -v perl pod/pod2man /tools/bin  
install -dv /tools/lib/perl5/5.8.8  
cp -Rv lib/* /tools/lib/perl5/5.8.8
```

Finally, create a necessary symlink:

```
ln -sv /tools/bin/perl /usr/bin
```

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

10.5. Linux-Headers-2.6.24.7

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 tarballs.

Install the header files that are common to all architectures:

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

10.5.2. Contents of Linux-Headers

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

Short Descriptions

```
/usr/include/{asm,asm-
generic,linux,mtd,rdma,scsi,sound,video}/
*.h
```

The Linux API headers

10.6. Man-pages-3.01

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. Glibc-2.7

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

10.7.1. Installation of Glibc



Note

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

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

The following patches merges all updates from the 2.7 Branch from the Glibc developers:

```
patch -Np1 -i ../glibc-2.7-branch_update-1A.patch
```

The following patch fixes an issue that can cause `localdef` to segfault:

```
patch -Np1 -i ../glibc-2.7-localedef_segfault-1.patch
```

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

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

Tell Glibc to install its libraries into `/lib`:

```
echo "slibdir=/lib" >> configparms
```

Prepare Glibc for compilation:

```
CFLAGS="-mtune=generic -g -O2" \
  ../glibc-2.7/configure --prefix=/usr \
  --disable-profile --enable-add-ons --enable-kernel=2.6.0 \
  --libexecdir=/usr/lib/glibc --libdir=/usr/lib
```

The meaning of the new configure option:

```
--libexecdir=/usr/lib/glibc
```

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

Compile the package:

```
make
```

**Important**

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

Test the results:

```
make -k check >glibc-check-log 2>&1 ; grep Error glibc-check-log
```

The Glibc 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 Glibc test suite is always expected to pass. However, in certain circumstances, some failures are unavoidable. If a test fails because of a missing program (or missing symbolic link), or a segfault, you will see an error code greater than 127 and the details will be in the log. More commonly, tests will fail with `Error 2` - for these, the contents of the corresponding `.out` file, e.g. `posix/annexc.out` may be informative. Here is a list of the most common issues:

- The *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 Glibc will complain about the absence of `/etc/ld.so.conf`. Prevent this warning with:

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

The install will finish by checking that everything is correctly installed. Unfortunately, it will test for a multilib installation. On `x86_64 Pure64` this means it will try to test the non-existent 32-bit loader which has a different name from the 64-bit loader (unlike on other 64-bit architectures). We fool it by creating a symlink to the real loader.

```
ln -sv ld-2.7.so /lib/ld-linux.so.2
```

Install the package:

```
make install
```

Now we can remove this symlink. We also need to correct the `/usr/bin/ldd` script - if you look at this, you will see it references not only the 32-bit linker, but also `/lib64` where it thinks the 64-bit linker is. The following `sed` will correct this:

```
rm -v /lib/ld-linux.so.2
cp -v /usr/bin/ldd{,.bak}
sed '/RTLDLIST/s%/ld-linux.so.2 /lib64%%' /usr/bin/ldd.bak >/usr/bin/ldd
```

Check the script to make sure the `sed` worked correctly, then delete the backup.

```
rm -v /usr/bin/ldd.bak
```

10.7.2. Internationalization

The locales that can make the system respond in a different language were not installed by the above command. Install them with:

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

To save time, an alternative to running the previous command (which generates and installs every locale listed in the `glibc-2.7/localedata/SUPPORTED` file) is to install only those locales that are wanted and needed. This can be achieved by using the **localedef** command. Information on this command is located in the `INSTALL` file in the Glibc source. However, there are a number of locales that are essential in order for the tests of future packages to pass, in particular, the `libstdc++` tests from GCC. The following instructions, instead of the `install-locales` target used above, will install the minimum set of locales necessary for the tests to run successfully:

```
mkdir -pv /usr/lib/locale
localedef -i cs_CZ -f UTF-8 cs_CZ.UTF-8
localedef -i de_DE -f ISO-8859-1 de_DE
localedef -i de_DE@euro -f ISO-8859-15 de_DE@euro
localedef -i en_HK -f ISO-8859-1 en_HK
localedef -i en_PH -f ISO-8859-1 en_PH
localedef -i en_US -f ISO-8859-1 en_US
localedef -i es_MX -f ISO-8859-1 es_MX
localedef -i fa_IR -f UTF-8 fa_IR
localedef -i fr_FR -f ISO-8859-1 fr_FR
localedef -i fr_FR@euro -f ISO-8859-15 fr_FR@euro
localedef -i it_IT -f ISO-8859-1 it_IT
localedef -i ja_JP -f EUC-JP ja_JP
```

Some locales installed by the **make localedata/install-locales** command above are not properly supported by some applications that are in the CLFS and BLFS books. 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 Glibc

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

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

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

passwd: files
group: files
shadow: files

hosts: files dns
networks: files

protocols: files
services: files
ethers: files
rpc: files

# End /etc/nsswitch.conf
EOF
```

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-linux-x86-64.so.2`) searches through `/lib` and `/usr/lib` for dynamic libraries that are needed by programs as they are run. However, if there are libraries in directories other than `/lib` and `/usr/lib`, these need to be added to the `/etc/ld.so.conf` file in order for the dynamic loader to find them. Two directories that are commonly known to contain additional libraries are `/usr/local/lib` and `/opt/lib`, so add those directories to the dynamic loader's search path.

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

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

/usr/local/lib
/opt/lib

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

10.7.5. Contents of Glibc

Installed programs: catchsegv, gencat, getconf, getent, iconv, iconvconfig, ldconfig, ldd, lddlibc4, locale, localedef, mtrace, nscd, pcprofiledump, pt_chown, rpcgen, rpcinfo, sln, sprof, tzselect, xtrace, zdump, and zic

Installed libraries: ld.so, libBrokenLocale.[a,so], libSegFault.so, libanl.[a,so], libbsd-compat.a, libc.[a,so], 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], libresolv.[a,so], librpcsvc.a, librt.[a,so], libthread_db.so, and libutil.[a,so]

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

mtrace Reads and interprets a memory trace file and displays a summary in human-readable format

nscd A daemon that provides a cache for the most common name service requests

pcprofiledump Dumps information generated by PC profiling

pt_chown A helper program for **grantpt** to set the owner, group and access permissions of a slave pseudo terminal

rpcgen Generates C code to implement the Remote Procedure Call (RPC) protocol

rpcinfo Makes an RPC call to an RPC server

sln A statically linked program that creates symbolic links

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
<code>ld.so</code>	The helper program for shared library executables
<code>libBrokenLocale</code>	Used by programs, such as Mozilla, to solve broken locales
<code>libSegFault</code>	The segmentation fault signal handler
<code>libanl</code>	An asynchronous name lookup library
<code>libbsd-compat</code>	Provides the portability needed in order to run certain Berkeley Software Distribution (BSD) programs under Linux
<code>libc</code>	The main C library
<code>libcrypt</code>	The cryptography library
<code>libdl</code>	The dynamic linking interface library
<code>libg</code>	A runtime library for <code>g++</code>
<code>libieee</code>	The Institute of Electrical and Electronic Engineers (IEEE) floating point library
<code>libm</code>	The mathematical library
<code>libmcheck</code>	Contains code run at boot
<code>libmemusage</code>	Used by memusage (included in <code>Glibc</code> , but not built in a base CLFS system as it has additional dependencies) to help collect information about the memory usage of a program
<code>libnsl</code>	The network services library
<code>libnss</code>	The Name Service Switch libraries, containing functions for resolving host names, user names, group names, aliases, services, protocols, etc.
<code>libpcprofile</code>	Contains profiling functions used to track the amount of CPU time spent in specific source code lines
<code>libpthread</code>	The POSIX threads library
<code>libresolv</code>	Contains functions for creating, sending, and interpreting packets to the Internet domain name servers
<code>librpcsvc</code>	Contains functions providing miscellaneous RPC services
<code>librt</code>	Contains functions providing most of the interfaces specified by the POSIX.1b Realtime Extension
<code>libthread_db</code>	Contains functions useful for building debuggers for multi-threaded programs
<code>libutil</code>	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-linux-x86-64.so.2]
```

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. Binutils-2.18

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

10.9.1. 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"
```

If the following message shows up, the environment is not set up for proper PTY operation:

```
The system has no more ptys.
Ask your system administrator to create more.
```

This issue needs to be resolved before running the test suites for Binutils and GCC.

The following patches merges all updates from the 2.18 Branch from the Binutils developers:

```
patch -Np1 -i ../binutils-2.18-branch_update-3.patch
```

Libiberty uses `gcc -print-multi-os-directory` to determine where to install libiberty.a. Because our temporary gcc is still multilib-aware, the command returns `../lib64` which causes the archive to be installed into `/usr/lib64`. The following sed will prevent this:

```
sed -i 's%\(^MULTIOSDIR = \).*%\1 ../lib%' libiberty/Makefile.in
```

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

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

Prepare Binutils for compilation:

```
../binutils-2.18/configure --prefix=/usr \
  --libdir=/usr/lib --enable-shared \
  --disable-multilib --enable-64-bit-bfd
```

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.

The ld test suite accesses /lib64/ld-linux-x86-64.so in some of the tests. The following symbolic link will allow this:

```
ln -sv /lib /lib64
```

Test the results:

```
make check
```

Now remove the temporary symlink:

```
rm -v /lib64
```

Now that the tests have completed, remove the symlinks in /usr/lib64 - they should be the only things in that directory, so we can remove it without force :

```
rm -v /usr/lib64/libstd*so*
rmdir -v /usr/lib64
```

Install the package:

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

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

```
cp -v ../binutils-2.18/include/libiberty.h /usr/include
```

10.9.2. Contents of Binutils

Installed programs: addr2line, ar, as, c++filt, gprof, ld, nm, objcopy, objdump, ranlib, readelf, size, strings, and strip

Installed libraries: libiberty.a, libbfd.[a,so], and libopcodes.[a,so]

Short Descriptions

addr2line Translates program addresses to file names and line numbers; given an address and the name of an executable, it uses the debugging information in the executable to determine which source file and line number are associated with the address

ar Creates, modifies, and extracts from archives

as An assembler that assembles the output of **gcc** into object files

c++filt Used by the linker to de-mangle C++ and Java symbols and to keep overloaded functions from clashing

gprof	Displays call graph profile data
ld	A linker that combines a number of object and archive files into a single file, relocating their data and tying up symbol references
nm	Lists the symbols occurring in a given object file
objcopy	Translates one type of object file into another
objdump	Displays information about the given object file, with options controlling the particular information to display; the information shown is useful to programmers who are working on the compilation tools
ranlib	Generates an index of the contents of an archive and stores it in the archive; the index lists all of the symbols defined by archive members that are relocatable object files
readelf	Displays information about ELF type binaries
size	Lists the section sizes and the total size for the given object files
strings	Outputs, for each given file, the sequences of printable characters that are of at least the specified length (defaulting to four); for object files, it prints, by default, only the strings from the initializing and loading sections while for other types of files, it scans the entire file
strip	Discards symbols from object files
libiberty	Contains routines used by various GNU programs, including getopt , obstack , strerror , strtol , and strtoul
libbfd	The Binary File Descriptor library
libopcodes	A library for dealing with opcodes—the “readable text” versions of instructions for the processor; it is used for building utilities like objdump .

10.10. GCC-4.2.4

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

10.10.1. Installation of GCC

The following patch enables fixes compilation errors with the kernel:

```
patch -Np1 -i ../gcc-4.2.4-PR31490-1.patch
```

Apply the following patch so that GCC links to /lib instead of /lib64:

```
patch -Np1 -i ../gcc-4.2.4-pure64-1.patch
```

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
```

GCC provides a **gccbug** script which detects at compile time whether `mktemp` is present, and hardcodes the result in a test. If `mktemp` is not found, the script will fall back to using less random names for temporary files. We will be installing `mktemp` later, so the following `sed` will simulate its presence:

```
sed -i 's/@have_mktemp_command@/yes/' gcc/gccbug.in
```

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

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

Prepare GCC for compilation:

```
../gcc-4.2.4/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
```

Compile the package:

```
make bootstrap
```



Important

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

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-4.2.4/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
```

10.10.2. 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> , <code>gcbug</code> , and <code>gcov</code>
Installed libraries:	<code>libgcc.a</code> , <code>libgcc_eh.a</code> , <code>libgcc_s.so</code> , <code>libgomp.[a,so]</code> , <code>libmudflap.[a,so]</code> , <code>libmudflapth.[a,so]</code> , <code>libssp.[a,so]</code> , <code>libstdc++.a</code> , <code>libstdc++.so</code> , and <code>libsupc++.a</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
gcbug	A shell script used to help create useful bug reports
gcov	A coverage testing tool; it is used to analyze programs to determine where optimizations will have the most effect
libgcc	Contains run-time support for gcc
libmudflap	The <code>libmudflap</code> libraries are used by GCC for instrumenting pointer and array dereferencing operations.
libstdc++	The standard C++ library
libsupc++	Provides supporting routines for the C++ programming language

10.11. Sed-4.1.5

The Sed package contains a stream editor.

10.11.1. Installation of Sed

Prepare Sed for compilation:

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

The meaning of the new configure option:

--enable-html

This option tells Sed to build and install its HTML documentation.

Compile the package:

```
make
```

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

Install the package:

```
make install
```

10.11.2. Contents of Sed

Installed program: sed

Short Descriptions

sed Filters and transforms text files in a single pass

10.12. E2fsprogs-1.40.4

The E2fsprogs package contains the utilities for handling the ext 2 file system. It also supports the ext 3 journaling file system.

10.12.1. 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-evms \
  --with-cc=gcc
```

The meaning of the configure options:

--with-root-prefix=""

Certain programs (such as the **e2fsck** program) are considered essential programs. When, for example, `/usr` is not mounted, these programs still need to be available. They belong in directories like `/lib` and `/sbin`. If this option is not passed to E2fsprogs' configure, the programs are installed into the `/usr` directory.

--enable-elf-shlibs

This creates the shared libraries which some programs in this package use.

--disable-evms

This disables the building of the Enterprise Volume Management System (EVMS) plugin. This plugin is not up-to-date with the latest EVMS internal interfaces and EVMS is not installed as part of a base CLFS system, so the plugin is not required. See the EVMS website at <http://evms.sourceforge.net/> for more information regarding EVMS.

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
```

10.12.2. Contents of E2fsprogs

Installed programs: badblocks, blkid, chattr, compile_et, debugfs, dumpe2fs, e2fsck, e2image, e2label, filefrag, findfs, fsck, fsck.ext2, fsck.ext3, logsave, lsattr, mk_cmds, mke2fs, mkfs.ext2, mkfs.ext3, mklost+found, resize2fs, tune2fs, uuid, and uuidgen.

Installed libraries: libblkid.[a,so], libcom_err.[a,so], libe2p.[a,so], libext2fs.[a,so], libss.[a,so], and libuuid.[a,so]

Short Descriptions

badblocks	Searches a device (usually a disk partition) for bad blocks
blkid	A command line utility to locate and print block device attributes
chattr	Changes the attributes of files on an <code>ext2</code> file system; it also changes <code>ext3</code> file systems, the journaling version of <code>ext2</code> file systems
compile_et	An error table compiler; it converts a table of error-code names and messages into a C source file suitable for use with the <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
e2fsck	Is used to check, and optionally repair <code>ext2</code> file systems and <code>ext3</code> file systems
e2image	Is used to save critical <code>ext2</code> file system data to a file
e2label	Displays or changes the file system label on the <code>ext2</code> file system present on a given device
filefrag	Reports on how badly fragmented a particular file might be
findfs	Finds a file system by label or Universally Unique Identifier (UUID)
fsck	Is used to check, and optionally repair, file systems
fsck.ext2	By default checks <code>ext2</code> file systems
fsck.ext3	By default checks <code>ext3</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> or <code>ext3</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
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
uudd	The <code>uudd</code> daemon is used by the UUID library to generate universally unique identifiers (UUIDs), especially time-based UUID's in a secure and guaranteed-unique fashion, even in the face of large numbers of threads trying to grab UUID's running on different CPU's.
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
<code>libblkid</code>	Contains routines for device identification and token extraction
<code>libcom_err</code>	The common error display routine
<code>libe2p</code>	Used by dumpe2fs , chattr , and lsattr
<code>libext2fs</code>	Contains routines to enable user-level programs to manipulate an <code>ext2</code> file system

<code>libss</code>	Used by debugfs
<code>libuuid</code>	Contains routines for generating unique identifiers for objects that may be accessible beyond the local system

10.13. Coreutils-6.9

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

10.13.1. Installation of Coreutils

Prevent Coreutils from installing binaries that will be installed by other packages:

```
patch -Np1 -i ../coreutils-6.9-suppress_uptime_kill_su-1.patch
```

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-6.9-uname-1.patch
```

The following command renames the internal implementation of `futimens` to `gl_futimens` as newer versions of Glibc provide an incompatible version:

```
sed -i "s/futimens/gl_&/" $(grep -lr futimens *)
```

Now prepare Coreutils for compilation:

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

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 check-root
```

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

```
src/su dummy -c "make RUN_EXPENSIVE_TESTS=yes check"
```

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,sticky,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
```

10.13.2. Contents of Coreutils

Installed programs: `[], base64, basename, cat, 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, mv, nice, nl, nohup, od, paste, pathchk, pinky, pr, printenv, printf, ptx, pwd, readlink, rm, rmdir, seq, sha1sum, sha224sum, sha256sum, sha384sum, sha512sum, shred, shuf, sleep, sort, split, stat, sticky, sum, sync, tac, tail, tee, test, touch, tr, true, tsort, tty, uname, unexpand, uniq, unlink, users, vdir, wc, who, whoami, and yes`

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
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 <code>/</code> 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
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
od	Dumps files in octal and other formats
paste	Merges the given files, joining sequentially corresponding lines side by side, separated by tab characters
pathchk	Checks if file names are valid or portable

pinky	Is a lightweight finger client; it reports some information about the given users
pr	Paginates and columnates files for printing
printenv	Prints the environment
printf	Prints the given arguments according to the given format, much like the C printf function
ptx	Produces a permuted index from the contents of the given files, with each keyword in its context
pwd	Reports the name of the current working directory
readlink	Reports the value of the given symbolic link
rm	Removes files or directories
rmdir	Removes directories if they are empty
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
stty	Sets or reports terminal line settings
sum	Prints checksum and block counts for each given file
sync	Flushes file system buffers; it forces changed blocks to disk and updates the super block
tac	Concatenates the given files in reverse
tail	Prints the last ten lines (or the given number of lines) of each given file
tee	Reads from standard input while writing both to standard output and to the given files
test or []	Compares values and checks file types
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
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

10.14. Iana-Etc-2.20

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

10.14.1. Installation of Iana-Etc

The following command converts the raw data provided by IANA into the correct formats for the `/etc/protocols` and `/etc/services` data files:

```
make
```

This package does not come with a test suite.

Install the package:

```
make install
```

10.14.2. 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

10.15. M4-1.4.10

The M4 package contains a macro processor.

10.15.1. 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
```

10.15.2. 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.

10.16. Bison-2.3

The Bison package contains a parser generator.

10.16.1. Installation of Bison

Prepare Bison for compilation:

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

The configure system causes bison to be built without support for internationalization of error messages if a **bison** program is not already in \$PATH. The following addition will correct this:

```
echo '#define YYENABLE_NLS 1' >> config.h
```

Compile the package:

```
make
```

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

Install the package:

```
make install
```

10.16.2. Contents of Bison

Installed programs: bison and yacc

Installed library: liby.a

Short Descriptions

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

10.17. Ncurses-5.6

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

10.17.1. Installation of Ncurses

The following patches merges all updates from the 5.6 Branch from the Ncurses developers:

```
patch -Np1 -i ../ncurses-5.6-branch_update-3.patch
```

Prepare Ncurses for compilation:

```
./configure --prefix=/usr --libdir=/lib \
  --with-shared --without-debug
```

Compile the package:

```
make
```

This package does not come with a test suite.

Install the package:

```
make install
```

Move the Ncurses static libraries to the proper location:

```
mv -v /lib/lib{panel,menu,form,ncurses,ncurses++,curses}.a /usr/lib
```

Create symlinks in /usr/lib:

```
rm -v /lib/lib{ncurses,menu,panel,form,curses}.so
ln -svf ../../lib/libncurses.so.5 /usr/lib/libcurses.so
ln -svf ../../lib/libncurses.so.5 /usr/lib/libncurses.so
ln -svf ../../lib/libmenu.so.5 /usr/lib/libmenu.so
ln -svf ../../lib/libpanel.so.5 /usr/lib/libpanel.so
ln -svf ../../lib/libform.so.5 /usr/lib/libform.so
```

Give the Ncurses dynamic libraries execute permissions:

```
chmod -v 755 /lib/lib{panel,menu,form,ncurses}.so.5.6
```

10.17.2. Contents of Ncurses

Installed programs: captinfo (link to tic), clear, infocmp, infotocap (link to tic), ncurses5-config, reset (link to tset), tack, tic, toe, tput, and tset

Installed libraries: libcurses.[a,so] (link to libncurses.[a,so]), libform.[a,so], libmenu.[a,so], libncurses++.a, libncurses.[a,so], and libpanel.[a,so]

Short Descriptions

captinfo 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
ncurses5-config	Provides configuration information for ncurses
reset	Reinitializes a terminal to its default values
tack	The terminfo action checker; it is mainly used to test the accuracy of an entry in the terminfo database
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
<code>libcurses</code>	A link to <code>libncurses</code>
<code>libncurses</code>	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
<code>libform</code>	Contains functions to implement forms
<code>libmenu</code>	Contains functions to implement menus
<code>libpanel</code>	Contains functions to implement panels

10.18. Procps-3.2.7

The Procps package contains programs for monitoring processes.

10.18.1. Installation of Procps

Compile the package:

```
make CC=gcc
```

This package does not come with a test suite.

Install the package:

```
make install
```

10.18.2. Contents of Procps

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

Installed library: libproc.so

Short Descriptions

free	Reports the amount of free and used memory (both physical and swap memory) in the system
kill	Sends signals to processes
pgrep	Looks up processes based on their name and other attributes
pkill	Signals processes based on their name and other attributes
pmap	Reports the memory map of the given process
ps	Lists the current running processes
pwdx	Reports the current working directory of a process
skill	Sends signals to processes matching the given criteria
slabtop	Displays detailed kernel 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

10.19. Libtool-1.5.26

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

10.19.1. 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
```

10.19.2. Contents of Libtool

Installed programs: libtool and libtoolize

Installed libraries: libltdl.[a,so]

Short Descriptions

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

10.20. Flex-2.5.35

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

10.20.1. 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
```

10.20.2. Contents of Flex

Installed programs: flex and lex

Installed library: libfl.a

Short Descriptions

flex	A tool for generating programs that recognize patterns in text; it allows for the versatility to specify the rules for pattern-finding, eradicating the need to develop a specialized program
lex	A script that runs flex in lex emulation mode
<code>libfl.a</code>	The <code>flex</code> library
<code>libfl_pic.a</code>	The <code>flex</code> library

10.21. IPRoute2-2.6.23

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

10.21.1. Installation of IPRoute2

The **arpd** binary included in this package is dependent on Berkeley DB. Because **arpd** is not a very common requirement on a base Linux system, remove the dependency on Berkeley DB by applying the **sed** command below. If the **arpd** binary is needed, instructions for compiling Berkeley DB can be found in the BLFS Book at http://cblfs.cross-lfs.org/index.php/Berkeley_DB.

```
sed -i '/^TARGETS/s@arpd@g' misc/Makefile
```

Compile the package:

```
make SBINDIR=/sbin
```

The meaning of the make option:

```
SBINDIR=/sbin
```

This ensures that the IPRoute2 binaries will install into `/sbin`. This is the correct location according to the FHS, because some of the IPRoute2 binaries are used by the CLFS-Bootscripts package.

This package does not come with a test suite.

Install the package:

```
make SBINDIR=/sbin install
```

10.21.2. Contents of IPRoute2

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

Short Descriptions

ctstat Connection status utility

ifcfg A shell script wrapper for the **ip** command

ifstat Shows the interface statistics, including the amount of transmitted and received packets by interface

ip The main executable. It has several different functions:

- ip link [device]** allows users to look at the state of devices and to make changes
- ip addr** allows users to look at addresses and their properties, add new addresses, and delete old ones
- ip neighbor** allows users to look at neighbor bindings and their properties, add new neighbor entries, and delete old ones
- ip rule** allows users to look at the routing policies and change them
- ip route** allows users to look at the routing table and change routing table rules
- ip tunnel** allows users to look at the IP tunnels and their properties, and change them
- ip maddr** allows users to look at the multicast addresses and their properties, and change them
- ip mroute** allows users to set, change, or delete the multicast routing
- ip monitor** allows users to continuously monitor the state of devices, addresses and routes

lnstat	Provides Linux network statistics. It is a generalized and more feature-complete replacement for the old rtstat program
nstat	Shows network statistics
routef	A component of ip route . This is for flushing the routing tables
routel	A component of ip route . This is for listing the routing tables
rtacct	Displays the contents of <code>/proc/net/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 queueing discipline tc class allows users to setup classes based on the queueing discipline scheduling tc estimator allows users to estimate the network flow into a network tc filter allows users to setup the QOS/COS packet filtering tc policy allows users to setup the QOS/COS policies

10.22. Perl-5.8.8

The Perl package contains the Practical Extraction and Report Language.

10.22.1. Installation of Perl

The following patch causes `makedepend.sh` to also remove `<command-line>` as well as `<command line>`:

```
patch -Np1 -i ../perl-5.8.8-gcc42_fix-1.patch
```

The following patch prevents perl from including `asm/page.h` which will no longer be available in headers provided by kernel 2.6.25 and forward:

```
patch -Np1 -i ../perl-5.8.8-no_page_h-1.patch
```

The following patch fixes known vulnerabilities in perl:

```
patch -Np1 -i ../perl-5.8.8-security_fixes-1.patch
```

The following sed causes `DynaLoader.a` to be built with `-fPIC` so it can be linked into a shared library later:

```
sed -i -e "s@pldflags='@pldflags=\"\${cccdlflags}\"@g" \
    -e "s@static_target='static'@static_target='static_pic'@g" Makefile.SH
```



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:

```
CC=gcc ./configure.gnu --prefix=/usr \
  -Dman1dir=/usr/share/man/man1 \
  -Dman3dir=/usr/share/man/man3 \
  -Dpager="/bin/less -isR" \
  -Dusetthreads
```

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.

-Dusethreads

This tells Perl to use threads.

Compile the package:

```
make
```

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

Install the package:

```
make install
```

10.22.2. Contents of Perl

Installed programs: a2p, c2ph, cpan, dprofpp, enc2xs, find2perl, h2ph, h2xs, instmodsh, libnetcfg, perl, perl5.8.8 (link to perl), perlbug, perlcc, perldoc, perlivp, piconv, pl2pm, pod2html, pod2latex, pod2man, pod2text, pod2usage, podchecker, podselect, prove, psed (link to s2p), pstruct (link to c2ph), s2p, splain, and xsubpp

Installed libraries: Several hundred which cannot all be listed here

Short Descriptions

a2p	Translates awk to Perl
c2ph	Dumps C structures as generated from cc -g -S
cpan	Shell script that provides a command interface to CPAN.pm
dprofpp	Displays Perl profile data
enc2xs	Builds a Perl extension for the Encode module from either Unicode Character Mappings or Tcl Encoding Files
find2perl	Translates find commands to Perl
h2ph	Converts .h C header files to .ph Perl header files
h2xs	Converts .h C header files to Perl extensions
libnetcfg	Can be used to configure the libnet
instmodsh	A shell script for examining installed Perl modules, and can even create a tarball from an installed module
perl	Combines some of the best features of C, sed , awk and sh into a single swiss-army-knife language
perl5.8.8	A hard link to perl
perlbug	Used to generate bug reports about Perl, or the modules that come with it, and mail them
perlcc	Generates executables from Perl programs
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
piconv	A Perl version of the character encoding converter iconv
pl2pm	A rough tool for converting Perl4 .pl files to Perl5 .pm modules

pod2html	Converts files from pod format to HTML format
pod2latex	Converts files from pod format to LaTeX format
pod2man	Converts pod data to formatted *roff input
pod2text	Converts pod data to formatted ASCII text
pod2usage	Prints usage messages from embedded pod docs in files
podchecker	Checks the syntax of pod format documentation files
podselect	Displays selected sections of pod documentation
prove	A command-line tool for running tests against Test::Harness
psed	A Perl version of the stream editor sed
pstruct	Dumps C structures as generated from cc -g -S stabs
s2p	Translates sed to Perl
splain	Is used to force verbose warning diagnostics in Perl
xsubpp	Converts Perl XS code into C code

10.23. Readline-5.2

The Readline package is a set of libraries that offers command-line editing and history capabilities.

10.23.1. 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-5.2-fixes-4.patch
```

Prepare Readline for compilation:

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

Compile the package:

```
make SHLIB_XLDFLAGS=-lncurses
```

The meaning of the make option:

```
SHLIB_XLDFLAGS=-lncurses
```

This option forces Readline to link against the `libncurses` library.

This package does not come with a test suite.

Install the package:

```
make install
```

Give Readline's dynamic libraries more appropriate permissions:

```
chmod -v 755 /lib/lib{readline,history}.so*
```

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.5 /usr/lib/libreadline.so
ln -svf ../../lib/libhistory.so.5 /usr/lib/libhistory.so
```

10.23.2. Contents of Readline

Installed libraries: `libhistory.[a,so]`, and `libreadline.[a,so]`

Short Descriptions

<code>libhistory</code>	Provides a consistent user interface for recalling lines of history
<code>libreadline</code>	Aids in the consistency of user interface across discrete programs that need to provide a command line interface

10.24. Zlib-1.2.3

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

10.24.1. Installation of Zlib

This patch will add `-fPIC` to our build and allow us to build a static and shared library at the same time:

```
patch -Np1 -i ../zlib-1.2.3-fPIC-1.patch
```

Prepare Zlib for compilation:

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

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
```

Now we fix the permissions on the static library:

```
chmod -v 644 /usr/lib/libz.a
```

10.24.2. Contents of Zlib

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

Short Descriptions

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

10.25. Autoconf-2.61

The Autoconf package contains programs for producing shell scripts that can automatically configure source code.

10.25.1. Installation of Autoconf

Prepare Autoconf for compilation:

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

Compile the package:

```
make
```

To test the results, issue: **make check**. 6 tests are skipped that use Automake. For full test coverage, Autoconf can be re-tested after Automake has been installed.

Install the package:

```
make install
```

10.25.2. Contents of Autoconf

Installed programs: autoconf, autoheader, autom4te, autoreconf, autoscan, autoupdate, and ifnames

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

10.26. Automake-1.10.1

The Automake package contains programs for generating Makefiles for use with Autoconf.

10.26.1. 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
```

10.26.2. Contents of Automake

Installed programs: acinstall, aclocal, aclocal-1.10, automake, automake-1.10, compile, config.guess, config.sub, depcomp, elisp-comp, install-sh, mdate-sh, missing, mkinstalldirs, py-compile, symlink-tree, and ylwrap

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-1.10	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-1.10	A hard link to automake
compile	A wrapper for compilers
config.guess	A script that attempts to guess the canonical triplet for the given build, host, or target architecture
config.sub	A configuration validation subroutine script
depcomp	A script for compiling a program so that dependency information is generated in addition to the desired output
elisp-comp	Byte-compiles Emacs Lisp code
install-sh	A script that installs a program, script, or data file
mdate-sh	A script that prints the modification time of a file or directory
missing	A script acting as a common stub for missing GNU programs during an installation
mkinstalldirs	A script that creates a directory tree

py-compile	Compiles a Python program
symlink-tree	A script to create a symlink tree of a directory tree
ylwrap	A wrapper for lex and yacc

10.27. Bash-3.2

The Bash package contains the Bourne-Again SHell.

10.27.1. Installation of Bash

If you downloaded the Bash documentation tarball and wish to install HTML documentation, issue the following commands:

```
tar -xvf ../bash-doc-3.2.tar.gz
```

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

```
patch -Np1 -i ../bash-3.2-fixes-8.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-3.2 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.

10.27.2. Contents of Bash

Installed programs: bash, bashbug, and sh (link to bash)

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

10.28. Bzip2-1.0.4

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**.

10.28.1. Installation of Bzip2

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
```

10.28.2. Contents of Bzip2

Installed programs: bunzip2 (link to bzip2), bzcat (link to bzip2), bzcmp, bzdiff, bzegrep, bzfgrep, bzgrep, bzip2, bzip2recover, bzless, and bzmores

Installed libraries: libbz2.a, libbz2.so (link to libbz2.so.1.0), libbz2.so.1.0 (link to libbz2.so.1.0.4), and libbz2.so.1.0.4

Short Descriptions

bunzip2	Decompresses bziped files
bzcat	Decompresses to standard output
bzcmp	Runs cmp on bziped files
bzdiff	Runs diff on bziped files
bzgrep	Runs grep on bziped files
bzegrep	Runs egrep on bziped files
bzfgrep	Runs fgrep 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

10.29. Diffutils-2.8.7

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

10.29.1. Installation of Diffutils

Prepare Diffutils for compilation:

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

Compile the package:

```
make
```

This package does not come with a test suite.

Install the package:

```
make install
```

10.29.2. 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

10.30. File-4.23

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

10.30.1. 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
```

10.30.2. 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

10.31. Findutils-4.2.32

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).

10.31.1. 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
```

10.31.2. Contents of Findutils

Installed programs: bigram, code, find, frcode, locate, updatedb, and xargs

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
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

10.32. Gawk-3.1.6

The Gawk package contains programs for manipulating text files.

10.32.1. Installation of Gawk

Prepare Gawk for compilation:

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

Now fix an issue that will cause the Gettext testsuite to fail:

```
echo '#define HAVE_LC_MESSAGES 1' >> config.h
```

Compile the package:

```
make
```

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

Install the package:

```
make install
```

10.32.2. Contents of Gawk

Installed programs: awk (link to gawk), gawk, gawk-3.1.6, great, igawk, pgawk, pgawk-3.1.6, and pwcats

Short Descriptions

awk	A link to gawk
gawk	A program for manipulating text files; it is the GNU implementation of awk
gawk-3.1.6	A hard link to gawk
great	Dumps the group database <code>/etc/group</code>
igawk	Gives gawk the ability to include files
pgawk	The profiling version of gawk
pgawk-3.1.6	Hard link to pgawk
pwcats	Dumps the password database <code>/etc/passwd</code>

10.33. Gettext-0.17

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.

10.33.1. 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
```

10.33.2. Contents of Gettext

Installed programs: autopoint, 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], and libgettextsrc.so

Short Descriptions

autopoint	Copies standard Gettext infrastructure files into a source package
envsubst	Substitutes environment variables in shell format strings
gettext	Translates a natural language message into the user's language by looking up the translation in a message catalog
gettext.sh	Primarily serves as a shell function library for gettext
gettextize	Copies all standard Gettext files into the given top-level directory of a package to begin internationalizing it
hostname	Displays a network hostname in various forms
msgattrib	Filters the messages of a translation catalog according to their attributes and manipulates the attributes
msgcat	Concatenates and merges the given .po files
msgcmp	Compares two .po files to check that both contain the same set of msgid strings
msgcomm	Finds the messages that are common to the given .po files
msgconv	Converts a translation catalog to a different character encoding
msgen	Creates an English translation catalog
msgexec	Applies a command to all translations of a translation catalog
msgfilter	Applies a filter to all translations of a translation catalog

msgfmt	Generates a binary message catalog from a translation catalog
msggrep	Extracts all messages of a translation catalog that match a given pattern or belong to some given source files
msginit	Creates a new <code>.po</code> 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 <code><string></code> strings and the <code><iostream></code> 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 <code>.po</code> files; this library is used when the standard applications shipped with Gettext (such as msgcomm , msgcmp , msgattrib , and msgen) 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

10.34. Grep-2.5.3

The Grep package contains programs for searching through files.

10.34.1. 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
```

10.34.2. 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

10.35. Groff-1.19.2

The Groff package contains programs for processing and formatting text.

10.35.1. 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
```

10.35.2. Contents of Groff

Installed programs: addftinfo, afmtodit, 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, refer, soelim, tbl, tfmtodit, troff, and zsoelim (link to soelim)

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
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
hptodit	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
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
soelim	Reads files and replaces lines of the form <code>.so file</code> by the contents of the mentioned <i>file</i>
tbl	Compiles descriptions of tables embedded within troff input files into commands that are understood by troff
tfmtodit	Creates a font file for use with groff -Tdvi
troff	Is highly compatible with Unix troff ; it should usually be invoked using the groff command, which will also run preprocessors and post-processors in the appropriate order and with the appropriate options

zsoelim

A link to **soelim**

10.36. Gzip-1.3.12

The Gzip package contains programs for compressing and decompressing files.

10.36.1. Installation of Gzip

The following command renames the internal implementation of futimens to gl_futimens as newer versions of Glibc provide an incompatible version:

```
sed -i "s/futimens/gl_&/" $(grep -lr futimens *)
```

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 utilities to /usr/bin to meet FHS compliance:

```
mv -v /bin/z{egrep,cmp,diff,fgrep,force,grep,less,more,new} /usr/bin
```

10.36.2. Contents of Gzip

Installed programs: gunzip, gzexe, gzip, uncompress, zcat, zcmp, zdiff, zegrep, zfgrep, zforce, zgrep, zless, zmore, and znew

Short Descriptions

gunzip	Decompresses gzipped files
gzexe	Creates self-decompressing executable files
gzip	Compresses the given files using Lempel-Ziv (LZ77) coding
uncompress	Decompresses compressed files
zcat	Decompresses the given gzipped files to standard output
zcmp	Runs cmp on gzipped files
zdiff	Runs diff on gzipped files
zegrep	Runs egrep on gzipped files
zfgrep	Runs fgrep on gzipped files
zforce	Forces a .gz extension on all given files that are gzipped files, so that gzip will not compress them again; this can be useful when file names were truncated during a file transfer
zgrep	Runs grep on gzipped files
zless	Runs less on gzipped files

zmore Runs **more** on gzipped files

znew Re-compresses files from **compress** format to **gzip** format— .Z to .gz

10.37. Inetutils-1.5

The Inetutils package contains programs for basic networking.

10.37.1. Installation of Inetutils

Not all programs that come with Inetutils will be installed. However, the Inetutils build system will insist on installing all the man pages anyway. The following patch will correct this situation:

```
patch -Np1 -i ../inetutils-1.5-no_server_man_pages-2.patch
```

Prepare Inetutils for compilation:

```
./configure --prefix=/usr --libexecdir=/usr/sbin \
  --sysconfdir=/etc --localstatedir=/var \
  --disable-logger --disable-syslogd \
  --disable-whois --disable-servers
```

The meaning of the configure options:

--disable-logger

This option prevents Inetutils from installing the **logger** program, which is used by scripts to pass messages to the System Log Daemon. Do not install it because Util-linux-ng installs a better version later.

--disable-syslogd

This option prevents Inetutils from installing the System Log Daemon, which is installed with the Sysklogd package.

--disable-whois

This option disables the building of the Inetutils **whois** client, which is out of date. Instructions for a better **whois** client are in the CBLFS book.

--disable-servers

This disables the installation of the various network servers included as part of the Inetutils package. These servers are deemed not appropriate in a basic CLFS system.

Compile the package:

```
make
```

This package does not come with a test suite.

Install the package:

```
make install
```

Move the **ping** and **ping6** programs to their FHS-compliant place:

```
mv -v /usr/bin/ping{,6} /bin
```

10.37.2. Contents of Inetutils

Installed programs: ftp, ifconfig, ping, ping6, rcp, rlogin, rsh, talk, telnet, and tftp

Short Descriptions

ftp	Is the file transfer protocol program
ifconfig	Utility for configuring network interfaces
ping	Sends echo-request packets and reports how long the replies take. This is the IPV4 version
ping6	Sends echo-request packets and reports how long the replies take. This is the IPV6 version
rcp	Performs remote file copy
rlogin	Performs remote login
rsh	Runs a remote shell
talk	Is used to chat with another user
telnet	An interface to the TELNET protocol
tftp	A trivial file transfer program

10.38. Kbd-1.13

The Kbd package contains key-table files and keyboard utilities.

10.38.1. Installation of Kbd

Prepare Kbd for compilation:

```
./configure --datadir=/lib/kbd
```

The meaning of the configure options:

--datadir

This option places the Kbd data and keymap files into `/lib/kbd`, as they are used by some of the scripts in the CLFS-Bootscripts package and must be on the root partition.

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,openvt,setfont} /bin
```

10.38.2. Contents of Kbd

Installed programs: `chvt`, `deallocvt`, `dumpkeys`, `fgconsole`, `getkeycodes`, `kbd_mode`, `kbdrate`, `loadkeys`, `loadunimap`, `mapscrn`, `openvt`, `psfaddtable` (link to `psfxtable`), `psfgettable` (link to `psfxtable`), `psfstrietable` (link to `psfxtable`), `psfxtable`, `resizecons`, `setfont`, `setkeycodes`, `setleds`, `setmetamode`, `showconsolefont`, `showkey`, `unicode_start`, and `unicode_stop`

Short Descriptions

<code>chvt</code>	Changes the foreground virtual terminal
<code>deallocvt</code>	Deallocates unused virtual terminals
<code>dumpkeys</code>	Dumps the keyboard translation tables
<code>fgconsole</code>	Prints the number of the active virtual terminal
<code>getkeycodes</code>	Prints the kernel scancode-to-keycode mapping table
<code>kbd_mode</code>	Reports or sets the keyboard mode
<code>kbdrate</code>	Sets the keyboard repeat and delay rates
<code>loadkeys</code>	Loads the keyboard translation tables
<code>loadunimap</code>	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
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

10.39. Less-418

The Less package contains a text file viewer.

10.39.1. 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
```

10.39.2. 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

10.40. Make-3.81

The Make package contains a program for compiling packages.

10.40.1. Installation of Make

Prepare Make for compilation:

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

Compile the package:

```
make
```

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

Install the package:

```
make install
```

10.40.2. 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

10.41. Man-1.6e

The Man package contains programs for finding and viewing man pages.

10.41.1. Installation of Man

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.

10.41.2. 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

10.42. Mktmp-1.5

The Mktmp package contains programs used to create secure temporary files in shell scripts.

10.42.1. Installation of Mktmp

Many scripts still use the deprecated **tempfile** program, which has functionality similar to **mktemp**. Patch Mktmp to include a **tempfile** wrapper:

```
patch -Np1 -i ../mktemp-1.5-add_tempfile-3.patch
```

Prepare Mktmp for compilation:

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

The meaning of the configure option:

```
--with-libc
```

This causes the **mktemp** program to use the *mkstemp* and *mkdtemp* functions from the system C library.

Compile the package:

```
make
```

This package does not come with a test suite.

Install the package:

```
make install
make install-tempfile
```

10.42.2. Contents of Mktmp

Installed programs: mktemp and tempfile

Short Descriptions

mktemp Creates temporary files in a secure manner; it is used in scripts

tempfile Creates temporary files in a less secure manner than **mktemp**; it is installed for backwards-compatibility

10.43. Module-Init-Tools-3.4

The Module-Init-Tools package contains programs for handling kernel modules in Linux kernels greater than or equal to version 2.5.47.

10.43.1. Installation of Module-Init-Tools

The tarball only contains sgml source for the manual pages. The following patch contains the result of processing this through **docbook2man** which we do not build as part of a basic clfs install :

```
patch -Np1 -i ../module-init-tools-3.4-manpages-1.patch
```

Issue the following commands to perform the tests



Note

note that the **make clean** command is required to clean up the source tree, otherwise the code built for testing will be installed, and the test version of **modprobe** does not function outside the test environment.

```
./configure &&  
make check &&  
make clean
```

Prepare Module-Init-Tools for compilation:

```
./configure --prefix=/ --enable-zlib
```

The meaning of the configure options:

--enable-zlib

This allows the Module-Init-Tools package to handle compressed kernel modules.

Compile the package:

```
make
```

Install the package:

```
make INSTALL=install install
```

The meaning of the make install parameter:

INSTALL=install

Normally, **make install** will not install the binaries if they already exist. This option overrides that behavior by calling **install** instead of using the default wrapper script.

10.43.2. Contents of Module-Init-Tools

Installed programs: depmod, generate-modprobe.conf, insmod, insmod.static, lsmod (link to insmod), modinfo, modprobe (link to insmod), and rmmod (link to insmod)

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

generate-modprobe.conf	Creates a modprobe.conf file from an existing 2.2 or 2.4 module setup
insmod	Installs a loadable module in the running kernel
insmod.static	A statically compiled version of insmod
lsmod	Lists currently loaded modules
modinfo	Examines an object file associated with a kernel module and displays any information that it can glean
modprobe	Uses a dependency file, created by depmod , to automatically load relevant modules
rmmod	Unloads modules from the running kernel

10.44. Patch-2.5.9

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

10.44.1. Installation of Patch

Prepare Patch for compilation:

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

Compile the package:

```
make
```

This package does not come with a test suite.

Install the package:

```
make install
```

10.44.2. 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.

10.45. Psmisc-22.6

The Psmisc package contains programs for displaying information about running processes.

10.45.1. 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
```

10.45.2. Contents of Psmisc

Installed programs: fuser, killall, 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
oldfuser	Reports the Process IDs (PIDs) of processes that use the given files or file systems
pstree	Displays running processes as a tree
pstree.x11	Same as pstree , except that it waits for confirmation before exiting

10.46. Shadow-4.1.2

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

10.46.1. 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.

Prepare Shadow for compilation:

```
./configure --libdir=/lib --sysconfdir=/etc --enable-shared \
--without-libpam --without-audit --without-selinux
```

The meaning of the configure options:

`--sysconfdir=/etc`

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

`--without-libpam`

Support for Linux-PAM is enabled by default in Shadow, however PAM is not installed on a base CLFS system, so this switch disables PAM support in Shadow. For instructions to install PAM and link Shadow to it, you can look at <http://cblfs.cross-lfs.org/index.php/Linux-PAM>.

`--without-audit`

Support for auditing is enabled by default, but a library that it needs is not installed in a base CLFS system. This switch disables auditing support.

`--without-selinux`

Support for selinux is enabled by default, but selinux is not built in a base CLFS system and configure will fail without this switch.

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

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

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 *MD5* 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@#MD5_CRYPT_ENAB.no@MD5_CRYPT_ENAB yes@' \
-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
```

10.46.2. 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.

10.46.3. Setting the root password

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

```
passwd root
```

10.46.4. Contents of Shadow

Installed programs: chage, chfn, chpasswd, chgpasswd, chsh, expiry, faillog, gpasswd, groupadd, groupdel, groupmod, grpck, grpconv, grpunconv, lastlog, login, logoutd, newgrp, newusers, nologin, passwd, pwck, pwconv, pwunconv, sg (link to newgrp), useradd, userdel, usermod, vigr (link to vipw), and vipw

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
chpasswd	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
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

10.47. Sysklogd-1.5

The Sysklogd package contains programs for logging system messages, such as those given by the kernel when unusual things happen.

10.47.1. Installation of Sysklogd

Compile the package:

```
make
```

This package does not come with a test suite.

Install the package:

```
make install
```

10.47.2. Configuring Sysklogd

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

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

auth,authpriv.*                -/var/log/auth.log
*.*;auth,authpriv.none         -/var/log/sys.log
daemon.*                       -/var/log/daemon.log
kern.*                         -/var/log/kern.log
mail.*                         -/var/log/mail.log
user.*                         -/var/log/user.log
*.info;mail.none;authpriv.none -/var/log/messages
*.emerg                        *

# log the bootscript output:
local2.*                       -/var/log/boot.log

# End /etc/syslog.conf
EOF
```

10.47.3. Contents of Sysklogd

Installed programs: klogd and syslogd

Short Descriptions

- klogd** A system daemon for intercepting and logging kernel messages
- syslogd** Logs the messages that system programs offer for logging. Every logged message contains at least a date stamp and a hostname, and normally the program's name too, but that depends on how trusting the logging daemon is told to be.

10.48. Sysvinit-2.86

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

10.48.1. Installation of Sysvinit

When run-levels are changed (for example, when halting the system), **init** sends termination signals to those processes that **init** itself started and that should not be running in the new run-level. While doing this, **init** outputs messages like “Sending processes the TERM signal” which seem to imply that it is sending these signals to all currently running processes. To avoid this misinterpretation, modify the source so that these messages read like “Sending processes started by init the TERM signal” instead:

```
sed -i 's@Sending processes@& started by init@g' \  
src/init.c
```

Compile the package:

```
make -C src clobber  
make -C src
```

Install the package:

```
make -C src install
```

10.48.2. 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

1:2345:respawn:/sbin/agetty -I '\033(K' tty1 9600
2:2345:respawn:/sbin/agetty -I '\033(K' tty2 9600
3:2345:respawn:/sbin/agetty -I '\033(K' tty3 9600
4:2345:respawn:/sbin/agetty -I '\033(K' tty4 9600
5:2345:respawn:/sbin/agetty -I '\033(K' tty5 9600
6:2345:respawn:/sbin/agetty -I '\033(K' tty6 9600

# 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.

10.48.3. Contents of Sysvinit

Installed programs: bootlogd, halt, init, killall5, last, lastb ([link to last](#)), mesg, mountpoint, pidof ([link to killall5](#)), poweroff ([link to halt](#)), reboot ([link to halt](#)), runlevel, shutdown, sulogin, telinit ([link to init](#)), utmpdump, and wall

Short Descriptions

bootlogd Logs boot messages to a log file

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
mountpoint	Tells you whether or not a directory is a mount point.
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
sulogin	Allows <i>root</i> to log in; it is normally invoked by init when the system goes into single user mode
telinit	Tells init which run-level to change to
utmpdump	Displays the content of the given login file in a more user-friendly format
wall	Writes a message to all logged-in users

10.49. Tar-1.20

The Tar package contains an archiving program.

10.49.1. Installation of Tar

Prepare Tar for compilation:

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

Compile the package:

```
make
```

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

Install the package:

```
make install
```

10.49.2. 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

10.50. Texinfo-4.11

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

10.50.1. Installation of Texinfo

Prepare Texinfo for compilation:

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

Compile the package:

```
make
```

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

Install the package:

```
make install
```

The Info documentation system uses a plain text file to hold its list of menu entries. The file is located at `/usr/share/info/dir`. Unfortunately, due to occasional problems in the Makefiles of various packages, it can sometimes get out of sync with the info pages installed on the system. If the `/usr/share/info/dir` file ever needs to be recreated, the following optional commands will accomplish the task:

```
cd /usr/share/info
rm dir
for f in *
do install-info $f dir 2>/dev/null
done
```

10.50.2. Contents of Texinfo

Installed programs: info, infokey, install-info, makeinfo, texi2dvi, texi2pdf, and texindex

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
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

10.51. Udev-124

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

10.51.1. Installation of Udev

Compile the package:

```
make EXTRAS="extras/ata_id extras/cdrom_id extras/collect \
  extras/edd_id extras/firmware extras/floppy extras/path_id extras/rule_generator
  extras/scsi_id extras/usb_id extras/volume_id" udevdir=/dev
```

The meaning of the make parameter:

```
EXTRAS="extras/ata_id    extras/cdrom_id    extras/collect    extras/edd_id
extras/firmware    extras/floppy    extras/path_id    extras/rule_generator
extras/scsi_id    extras/usb_id    extras/volume_id"
```

This builds the helper applications that are used with udev. The helper programs assist in correct handling of devices.

If you want to run the testsuite, you need to change a hardcoded reference to the `test` program:

```
sed -i 's@/usr/bin/test@/bin/test@' test/udev-test.pl
```

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

Install the package:

```
make EXTRAS="extras/ata_id extras/cdrom_id extras/collect \
  extras/edd_id extras/firmware extras/floppy extras/path_id extras/rule_generator
  extras/scsi_id extras/usb_id extras/volume_id" DESTDIR=/ udevdir=/dev install
```

The meaning of the make parameter:

```
DESTDIR=/
```

This prevents the Udev build process from killing any `udevd` processes that may be running on the system.

Install documentation:

```
install -v -m644 -D docs/writing_udev_rules/index.html \
  /usr/share/doc/udev-124/index.html
```

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

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

10.51.2. Contents of Udev

Installed programs:	udevadm, udevd, udevinfo (symlink to udevadm), udevsettle (symlink to udevadm)
Installed library:	libvolume_id
Installed directory:	/etc/udev, /lib/udev

Short Descriptions

`udevadm` This program combines `udevinfo`, `udevtrigger`, `udevsettle`, `udevcontrol`, `udevmonitor`, and `udevtest` into one command.

udevadm control	Configures a number of options for the running udev daemon, such as the log level. (Symlink to udevadm)
udev	A daemon that reorders hotplug events before submitting them to udev , thus avoiding various race conditions
udevinfo	Allows users to query the udev database for information on any device currently present on the system; it also provides a way to query any device in the <code>sysfs</code> tree to help create udev rules. (Symlink to udevadm)
udevadm monitor	Prints the event received from the kernel and the event which udev sends out after rule processing
udevsettle	Watches the Udev event queue and exits if all current uevents have been handled. (Symlink to udevadm)
udevadm test	Simulates a udev run for the given device, and prints out the name of the node the real udev would have created or the name of the renamed network interface
udevadm trigger	Walks the <code>sysfs</code> tree for devices that need to be added to the system.
ata_id	Provides Udev with a unique string and additional information (uuid, label) for an ATA drive
cdrom_id	Print the capabilities of a CDROM or DVDROM drive.
create_floppy_devices	Creates all possible floppy devices based on the CMOS type
dasd_id	Read the label from an s390 block device.
edd_id	Identify x86 disk drives from Enhanced Disk Drive calls.
firmware.sh	Script to load firmware for a device
path_id	Provide the shortest possible unique hardware path to a device
scsi_id	Retrieve or generate a unique SCSI identifier.
usb_id	Identify a USB block device.
vol_id	Probe filesystem type and read label and uuid.
<code>libvolume_id</code>	This library provides programs such as vol_id with a filesystems type, label, and uuid from a device node or a image.
<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.

10.52. Util-linux-ng-2.14

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

10.52.1. 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 's@etc/adjtime@var/lib/hwclock/adjtime@g' \
    hwclock/hwclock.c
mkdir -pv /var/lib/hwclock
```

10.52.2. Installation of Util-linux-ng

Prepare Util-linux-ng for compilation:

```
./configure --enable-partx \
    --disable-wall --enable-write
```

Compile the package:

```
make
```

This package does not come with a test suite.

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
```

10.52.3. Contents of Util-linux-ng

Installed programs: `addpart`, `agetty`, `blockdev`, `cal`, `cdisk`, `chkdupexe`, `col`, `colcrt`, `colrm`, `column`, `ctrlaltdel`, `cytune`, `ddate`, `delpart`, `dmesg`, `fdformat`, `fdisk`, `flock`, `fsck.cramfs`, `fsck.minix`, `getopt`, `hexdump`, `hwclock`, `ionice`, `ipcrm`, `ipcs`, `isozsize`, `line`, `logger`, `look`, `losetup`, `mcookie`, `mkfs`, `mkfs.bfs`, `mkfs.cramfs`, `mkfs.minix`, `mkswap`, `more`, `mount`, `namei`, `partx`, `pg`, `pivot_root`, `readprofile`, `rename`, `renice`, `rev`, `rtcwake`, `script`, `scriptreplay`, `setarch`, `setuid`, `setterm`, `sfdisk`, `swapoff` (link to `swapon`), `swapon`, `tailf`, `taskset`, `tunelp`, `ul`, `umount`, `whereis`, and `write`

Short Descriptions

addpart	Notifies the kernel of a new partition
agetty	Opens a tty port, prompts for a login name, and then invokes the login program
blockdev	Allows users to call block device ioctls from the command line
cal	Displays a simple calendar

cdisk	Manipulates the partition table of the given device
chkdupexe	Finds duplicate executables
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
fdformat	Low-level formats a floppy disk
fdisk	Manipulates the partition table of the given device
flock	Acquires a file lock and then executes a command with the lock held
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
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
ipcrm	Removes the given Inter-Process Communication (IPC) resource
ipcs	Provides IPC status information
isosize	Reports the size of an iso9660 file system
line	Copies a single 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
mcookie	Generates magic cookies (128-bit random hexadecimal numbers) for xauth
mkfs	Builds a file system on a device (usually a hard disk partition)
mkfs.bfs	Creates a Santa Cruz Operations (SCO) bfs file system
mkfs.cramfs	Creates a cramfs file system
mkfs.minix	Creates a Minix file system
mkswap	Initializes the given device or file to be used as a swap area
more	A filter for paging through text one screen at a time
mount	Attaches the file system on the given device to a specified directory in the file-system tree

namei	Shows the symbolic links in the given pathnames
partx	Tells the kernel about the presence and numbering of on-disk partitions
pg	Displays a text file one screen full at a time
pivot_root	Makes the given file system the new root file system of the current process
readprofile	Reads kernel profiling information
rename	Renames the given files, replacing a given string with another
renice	Alters the priority of running processes
rev	Reverses the lines of a given file
rtcwake	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
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
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
whereis	Reports the location of the binary, source, and man page for the given command
write	Sends a message to the given user <i>if</i> that user has not disabled receipt of such messages

10.53. Vim-7.1

The Vim package contains a powerful text editor.



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.

10.53.1. Installation of Vim

First, unpack both `vim-7.1.tar.bz2` and (optionally) `vim-7.1-lang.tar.gz` archives into the same directory.

The following patches merges all updates from the 7.1 Branch from the Vim developers:

```
patch -Np1 -i ../vim-7.1-branch_update-2.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 Core 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-7.1`, making it consistent with the location of documentation for other packages:

```
ln -sv ../vim/vim71/doc /usr/share/doc/vim-7.1
```

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 the CBLFS book at <http://cblfs.cross-lfs.org/index.php/Vim>.

10.53.2. Configuring Vim

By default, **vim** runs in vi-incompatible mode. This may be new to users who have used other editors in the past. The “nocompatible” setting is included below to highlight the fact that a new behavior is being used. It also reminds those who would change to “compatible” mode that it should be the first setting in the configuration file. This is necessary because it changes other settings, and overrides must come after this setting. Create a default **vim** configuration file by running the following:

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

set nocompatible
set backspace=2
syntax on
if (&term == "iterm") || (&term == "putty")
    set background=dark
endif

" End /etc/vimrc
EOF
```

The *set nocompatible* 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'
```

10.53.3. 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), vimm, vimspell.sh, vimtutor, and xxd

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

10.54. Bin86-0.16.17

The Bin86 package contains an assembler and linker to produce 16 or 32-bit 8086 machine code.

10.54.1. Installation of Bin86

We are building Bin86 because it is required to compile Lilo, and at the moment no other bootloader builds and runs on a pure64 system. If your machine has multiple systems (i.e. x86 or multilib) you may prefer to use the bootloader from those systems, such as GRUB.

This patch updates Bin86 to compile on x86_64:

```
patch -Np1 -i ../bin86-0.16.17-x86_64-1.patch
```

Compile the package:

```
make
```

Install the package:

```
make PREFIX=/usr install
```

10.54.2. Contents of Bin86

Installed programs: as86, ld86, nm86, objdump86, size86

Short Descriptions

as86	An 8086 assembler
ld86	A linker for the object files produced by as86
nm86	Symbolic link to objdump86
objdump86	A utility to examine and report on the output from as86 and ld86
size86	Symbolic link to objdump86

10.55. Lilo-22.8

The Lilo package contains the Linux Loader, a bootloader.

We have chosen to use Lilo because at the moment no other bootloader builds and runs on a pure64 system. If your machine has multiple systems (i.e. x86 or multilib) you may prefer to use the bootloader from the other system, such as GRUB.

10.55.1. Installation of Lilo

Compile the package:

```
make
```

Install the package:

```
make install
```

At the end of the installation the make install process will print a message stating that /sbin/lilo has to be executed to complete the update. Don't do this as it has no use. The /etc/lilo.conf isn't present yet. We will complete the installation of Lilo in chapter 12.

10.55.2. Contents of Lilo

Installed programs: diag1.img, lilo, mkrescue, keytab-lilo.pl

Short Descriptions

diag1.img	A diagnostic disk boot image.
lilo	Lilo installs the Linux boot loader which is used to start a Linux system.
mkrescue	A script to make a bootable floppy or CD image using the default settings from the configuration file.
keytab-lilo.pl	A perl script to create a keyboard translation table to allow the bootloader to process keystrokes to match your keyboard layout.

10.56. 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.

10.57. 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/bsd-init.txt>. 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. CLFS-Bootscripts-1.1.0

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

11.2.1. Installation of CLFS-Bootscripts

Install the package:

```
make install-bootscripts
```

11.2.2. Contents of CLFS-Bootscripts

Installed scripts: checkfs, cleanfs, console, functions, halt, ifdown, ifup, localnet, mountfs, mountkernfs, network, rc, reboot, sendsignals, setclock, static, swap, sysklogd, and template.

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

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. Device and Module Handling on a CLFS System

In *Installing Basic System Software*, we installed the Udev package. Before we go into the details regarding how this works, a brief history of previous methods of handling devices is in order.

Linux systems in general traditionally use a static device creation method, whereby a great many device nodes are created under `/dev` (sometimes literally thousands of nodes), regardless of whether the corresponding hardware devices actually exist. This is typically done via a **MAKEDEV** script, which contains a number of calls to the **mknod** program with the relevant major and minor device numbers for every possible device that might exist in the world.

Using the Udev method, only those devices which are detected by the kernel get device nodes created for them. Because these device nodes will be created each time the system boots, they will be stored on a `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.

11.4.1. History

In February 2000, a new filesystem called `devfs` was merged into the 2.3.46 kernel and was made available during the 2.4 series of stable kernels. Although it was present in the kernel source itself, this method of creating devices dynamically never received overwhelming support from the core kernel developers.

The main problem with the approach adopted by `devfs` was the way it handled device detection, creation, and naming. The latter issue, that of device node naming, was perhaps the most critical. It is generally accepted that if device names are allowed to be configurable, then the device naming policy should be up to a system administrator, not imposed on them by any particular developer(s). The `devfs` file system also suffers from race conditions that are inherent in its design and cannot be fixed without a substantial revision to the kernel. It 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.

11.4.2. Udev Implementation

11.4.2.1. Sysfs

The `sysfs` filesystem was mentioned briefly above. One may wonder how `sysfs` knows about the devices present on a system and what device numbers should be used for them. Drivers that have been compiled into the kernel directly register their objects with `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.

11.4.2.2. Udev 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 Udev 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.

11.4.2.3. Device Node Creation

To obtain the right major and minor number for a device, Udev 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 Udev rules configuration files are available in `/usr/share/doc/udev-124/index.html`

11.4.2.4. Module Loading

Device drivers compiled as modules may have aliases built into them. Aliases are visible in the output of the **modinfo** program and are usually related to the bus-specific identifiers of devices supported by a module. For example, the `snd-fm801` driver supports PCI devices with vendor ID 0x1319 and device ID 0x0801, and has an alias of “pci:v00001319d00000801sv*sd*bc04sc01i*”. For most devices, the bus driver exports the alias of the driver that would handle the device via `sysfs`. E.g., the `/sys/bus/pci/devices/0000:00:0d.0/modalias` file might contain the string “pci:v00001319d00000801sv00001319sd00001319bc04sc01i00”. The rules that CLFS installs 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.

11.4.2.5. Handling Hotpluggable/Dynamic Devices

When you plug in a device, such as a Universal Serial Bus (USB) MP3 player, the kernel recognizes that the device is now connected and generates a uevent. This uevent is then handled by **udev** as described above.

11.4.3. Problems with Loading Modules and Creating Devices

There are a few possible problems when it comes to automatically creating device nodes.

11.4.3.1. A kernel module is not loaded automatically

Udev will only load a module if it has a bus-specific alias and the bus driver properly exports the necessary aliases to `sysfs`. In other cases, one should arrange module loading by other means. With Linux-2.6.24.7, Udev is known to load properly-written drivers for INPUT, IDE, PCI, USB, SCSI, SERIO and FireWire devices.

To determine if the device driver you require has the necessary support for Udev, run **modinfo** with the module name as the argument. Now try locating the device directory under `/sys/bus` and check whether there is a `modalias` file there.

If the `modalias` file exists in `sysfs`, the driver supports the device and can talk to it directly, but doesn't have the alias, it is a bug in the driver. Load the driver without the help from Udev and expect the issue to be fixed later.

If there is no `modalias` file in the relevant directory under `/sys/bus`, this means that the kernel developers have not yet added `modalias` support to this bus type. With Linux-2.6.24.7, this is the case with ISA busses. Expect this issue to be fixed in later kernel versions.

Udev is not intended to load “wrapper” drivers such as `snd-pcm-oss` and non-hardware drivers such as `loop` at all.

11.4.3.2. A kernel module is not loaded automatically, and Udev is not intended to load it

If the “wrapper” module only enhances the functionality provided by some other module (e.g., `snd-pcm-oss` enhances the functionality of `snd-pcm` by making the sound cards available to OSS applications), configure **modprobe** to load the wrapper after Udev loads the wrapped module. To do this, add 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.

11.4.3.3. Udev 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.

11.4.3.4. Udev creates a device incorrectly, or makes a wrong symlink

This usually happens if a rule unexpectedly matches a device. For example, a poorly-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.

11.4.3.5. Udev rule works unreliably

This may be another manifestation of the previous problem. If not, and your rule uses `sysfs` attributes, it may be a kernel timing issue, to be fixed in later kernels. For now, you can work around it by creating a rule that waits for the used `sysfs` attribute and appending it to the `/etc/udev/rules.d/10-wait_for_sysfs.rules` file. Please notify the CLFS Development list if you do so and it helps.

11.4.3.6. Udev does not create a device

Further text assumes that the driver is built statically into the kernel or already loaded as a module, and that you have already checked that Udev doesn't create a misnamed device.

Udev has no information needed to create a device node if a kernel driver does not export its data to `sysfs`. This is most common with third party drivers from outside the kernel tree. Create a static device node in `/lib/udev/devices` with the appropriate major/minor numbers (see the file `devices.txt` inside the kernel documentation or the documentation provided by the third party driver vendor). The static device node will be copied to `/dev` by the **S10udev** bootscript.

11.4.3.7. Device naming order changes randomly after rebooting

This is due to the fact that Udev, by design, handles uevents and loads modules in parallel, and thus in an unpredictable order. This will never be “fixed”. You should not rely upon the kernel device names being stable. Instead, create your own rules that make symlinks with stable names based on some stable attributes of the device, such as a serial number or the output of various `*_id` utilities installed by Udev. See Section 11.12, “Creating custom symlinks to devices” and Section 11.13, “Configuring the network Script” for examples.

11.4.4. Useful Reading

Additional helpful documentation is available at the following sites:

- A Userspace Implementation of `devfs`
http://www.kroah.com/linux/talks/ols_2003_udev_paper/Reprint-Kroah-Hartman-OLS2003.pdf
- udev FAQ
<http://www.kernel.org/pub/linux/utils/kernel/hotplug/udev-FAQ>
- The `sysfs` Filesystem
<http://www.kernel.org/pub/linux/kernel/people/mochel/doc/papers/ols-2005/mochel.pdf>

11.5. 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/time.txt>. It explains issues such as time zones, UTC, and the TZ environment variable.

11.6. Configuring the Linux Console

This section discusses how to configure the **console** 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 **console** script reads the `/etc/sysconfig/console` 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/console` 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 `/lib/kbd` directory for valid keymaps and screen fonts. Read `loadkeys(1)` and `setfont(8)` to determine the correct arguments for these programs. Once decided, create the configuration file with the following command:

```
cat >/etc/sysconfig/console <<"EOF"
KEYMAP="[arguments for loadkeys]"
FONT="[arguments for setfont]"
EOF
```

For example, for Spanish users who also want to use the Euro character (accessible by pressing AltGr+E), the following settings are correct:

```
cat >/etc/sysconfig/console <<"EOF"
KEYMAP="es euro2"
FONT="lat9-16 -u iso01"
EOF
```



Note

The FONT line above is correct only for the ISO 8859-15 character set. If using ISO 8859-1 and, therefore, a pound sign instead of Euro, the correct FONT line would be:

```
FONT="lat1-16"
```

If the KEYMAP or FONT variable is not set, the **console** initscript will not run the corresponding program.

In some keymaps, the Backspace and Delete keys send characters different from ones in the default keymap built into the kernel. This confuses some applications. For example, Emacs displays its help (instead of erasing the character before the cursor) when Backspace is pressed. To check if the keymap in use is affected (this works only for i386 keymaps):

```
zgrep '\W14\W' [/path/to/your/keymap]
```

If the keycode 14 is Backspace instead of Delete, create the following keymap snippet to fix this issue:

```
mkdir -pv /etc/kbd && cat > /etc/kbd/bs-sends-del <<"EOF"
    keycode 14 = Delete Delete Delete Delete
    alt keycode 14 = Meta_Delete
    altgr alt keycode 14 = Meta_Delete
    keycode 111 = Remove
    altgr control keycode 111 = Boot
    control alt keycode 111 = Boot
    altgr control alt keycode 111 = Boot
EOF
```

Tell the **console** script to load this snippet after the main keymap:

```
cat >>/etc/sysconfig/console <<"EOF"
KEYMAP_CORRECTIONS="/etc/kbd/bs-sends-del"
EOF
```

11.7. Configuring the syslogd script

The `syslogd` script invokes the `syslogd` program with the `-m 0` option. This option turns off the periodic timestamp mark that `syslogd` writes to the log files every 20 minutes by default. If you want to turn on this periodic timestamp mark, edit the `syslogd` script and make the changes accordingly. See `man syslogd` for more information.

11.8. Creating the /etc/inputrc File

The `/etc/inputrc` file deals with mapping the keyboard for specific situations. This file is the start-up file used by Readline — the input-related library — used by Bash and most other shells.

Most people do not need user-specific keyboard mappings so the command below creates a global `/etc/inputrc` used by everyone who logs in. If you later decide you need to override the defaults on a per-user basis, you can create a `.inputrc` file in the user's home directory with the modified mappings.

For more information on how to edit the `inputrc` file, see `info bash` under the *Readline Init File* section. `info readline` is also a good source of information.

Below is a generic global `inputrc` along with comments to explain what the various options do. Note that comments cannot be on the same line as commands. Create the file using the following command:

```
cat > /etc/inputrc << "EOF"
# Begin /etc/inputrc
# Modified by Chris Lynn <roryo@roryo.dynup.net>

# Allow the command prompt to wrap to the next line
set horizontal-scroll-mode Off

# Enable 8bit input
set meta-flag On
set input-meta On
```

```

# Turns off 8th bit stripping
set convert-meta Off

# Keep the 8th bit for display
set output-meta On

# none, visible or audible
set bell-style none

# All of the following map the escape sequence of the
# value contained inside the 1st argument to the
# readline specific functions

"\eOd": backward-word
"\eOc": forward-word

# for linux console
"\e[1~": beginning-of-line
"\e[4~": end-of-line
"\e[5~": beginning-of-history
"\e[6~": end-of-history
"\e[3~": delete-char
"\e[2~": quoted-insert

# for xterm
"\eOH": beginning-of-line
"\eOF": end-of-line

# for Konsole
"\e[H": beginning-of-line
"\e[F": end-of-line

# End /etc/inputrc
EOF

```

11.9. The Bash Shell Startup Files

The shell program **/bin/bash** (hereafter referred to as “the shell”) uses a collection of startup files to help create an environment to run in. Each file has a specific use and may affect login and interactive environments differently. The files in the `/etc` directory provide global settings. If an equivalent file exists in the home directory, it may override the global settings.

An interactive login shell is started after a successful login, using **/bin/login**, by reading the `/etc/passwd` file. An interactive non-login shell is started at the command-line (e.g., `[prompt]$/bin/bash`). A non-interactive shell is usually present when a shell script is running. It is non-interactive because it is processing a script and not waiting for user input between commands.

For more information, see **info bash** under the *Bash Startup Files and Interactive Shells* section.

The files `/etc/profile` and `~/ .bash_profile` are read when the shell is invoked as an interactive login shell.

The base `/etc/profile` below sets some environment variables necessary for native language support. Setting them properly results in:

- The output of programs translated into the native language
- Correct classification of characters into letters, digits and other classes. This is necessary for **bash** to properly accept non-ASCII characters in command lines in non-English locales
- The correct alphabetical sorting order for the country
- Appropriate default paper size
- Correct formatting of monetary, time, and date values

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_GB.iso88591” in our example).

```
LC_ALL=[locale name] locale charmap
```

For the “en_GB.iso88591” locale, the above command will print:

```
ISO-8859-1
```

This results in a final locale setting of “en_GB.ISO-8859-1”. It is important that the locale found using the heuristic above is tested prior to it being added to the Bash startup files:

```
LC_ALL=[locale name] locale country
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
```



Note

The "C" (default) and "en_US" (the recommended one for United States English users) locales are different.

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. More complex cases (including UTF-8 based locales) require additional steps and additional patches because many applications tend to not work properly under such conditions. These steps and patches are not included in the CLFS book and such locales are not yet supported by CLFS.

11.10. 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.

11.11. 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
```

11.12. Creating custom symlinks to devices

11.12.1. 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}=="5V01306DM00190", SYMLINK+="cdrom1 dvd"

EOF
```



Note

Although the examples in this book work properly, be aware that Udev does not recognize the backslash for line continuation. If modifying Udev rules with an editor, be sure to leave each rule on one physical line.

This way, the symlinks will stay correct even if you move the drives to different positions on the IDE bus, but the `/dev/cdrom` symlink won't be created if you replace the old SAMSUNG CD-ROM with a new drive.

The `SUBSYSTEM=="block"` key is needed in order to avoid matching SCSI generic devices. Without it, in the case with SCSI CD-ROMs, the symlinks will sometimes point to the correct `/dev/srX` devices, and sometimes to `/dev/sgX`, which is wrong.

The second approach yields:

```
cat >/etc/udev/rules.d/82-cdrom.rules << EOF

# Custom CD-ROM symlinks
SUBSYSTEM=="block", ENV{ID_TYPE}=="cd", \
    ENV{ID_PATH}=="pci-0000:00:07.1-ide-0:1", SYMLINK+="cdrom"
SUBSYSTEM=="block", ENV{ID_TYPE}=="cd", \
    ENV{ID_PATH}=="pci-0000:00:07.1-ide-1:1", SYMLINK+="cdrom1 dvd"

EOF
```

This way, the symlinks will stay correct even if you replace drives with different models, but place them to the old positions on the IDE bus. The `ENV{ID_TYPE}=="cd"` key makes sure that the symlink disappears if you put something other than a CD-ROM in that position on the bus.

Of course, it is possible to mix the two approaches.

11.12.2. Dealing with duplicate devices

As explained in Section 11.4, “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 Section 11.13, “Configuring the network Script”, 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 Udev rules can be found in `/usr/share/doc/udev-124/index.html`.

11.13. Configuring the network Script

This section only applies if a network card is to be configured.

If a network card will not be used, there is likely no need to create any configuration files relating to network cards. If that is the case, remove the network symlinks from all run-level directories (`/etc/rc.d/rc*.d`).

11.13.1. Creating Network Interface Configuration Files

Which interfaces are brought up and down by the network script depends on the files and directories in the `/etc/sysconfig/network-devices` hierarchy. This directory should contain a sub-directory for each interface to be configured, such as `ifconfig.xyz`, where “xyz” is a network interface name. Inside this directory would be files defining the attributes to this interface, such as its IP address(es), subnet masks, and so forth.

The following command creates a sample `ipv4` file for the `eth0` device:

```
cd /etc/sysconfig/network-devices &&
mkdir -v ifconfig.eth0 &&
cat > ifconfig.eth0/ipv4 << "EOF"
ONBOOT=yes
SERVICE=ipv4-static
IP=192.168.1.1
GATEWAY=192.168.1.2
PREFIX=24
BROADCAST=192.168.1.255
EOF
```

The values of these variables must be changed in every file to match the proper setup. If the `ONBOOT` variable is set to “yes” the network script will bring up the Network Interface Card (NIC) during booting of the system. If set to anything but “yes” the NIC will be ignored by the network script and not be brought up.

The `SERVICE` variable defines the method used for obtaining the IP address. The 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. This is commonly used for Dynamic Host Configuration Protocol (DHCP), which is addressed in the BLFS book.

The `GATEWAY` variable should contain the default gateway IP address, if one is present. If not, then comment out the variable entirely.

The `PREFIX` variable needs to contain the number of bits used in the subnet. Each octet in an IP address is 8 bits. If the subnet's netmask is `255.255.255.0`, then it is using the first three octets (24 bits) to specify the network number. If the netmask is `255.255.255.240`, it would be using the first 28 bits. Prefixes longer than 24 bits are commonly used by DSL and cable-based Internet Service Providers (ISPs). In this example (`PREFIX=24`), the netmask is `255.255.255.0`. Adjust the `PREFIX` variable according to your specific subnet.

11.13.2. Creating the `/etc/resolv.conf` File

If the system is going to be connected to the Internet, it will need some means of Domain Name Service (DNS) name resolution to resolve Internet domain names to IP addresses, and vice versa. This is best achieved by placing the IP address of the DNS server, available from the ISP or network administrator, into `/etc/resolv.conf`. Create the file by running the following:

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

domain [Your Domain Name]
nameserver [IP address of your primary nameserver]
nameserver [IP address of your secondary nameserver]

# End /etc/resolv.conf
EOF
```

Replace `[IP address of the nameserver]` with the IP address of the DNS most appropriate for the setup. There will often be more than one entry (requirements demand secondary servers for fallback capability). If you only need or want one DNS server, remove the second `nameserver` line from the file. The IP address may also be a router on the local network.

Chapter 12. Making the CLFS System Bootable

12.1. 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.

12.2. 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=4,mode=620          0     0
shm           /dev/shm     tmpfs   defaults                 0     0
# End /etc/fstab
EOF
```

Replace `[xxx]`, `[yyy]`, and `[fff]` with the values appropriate for the system, for example, `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.

12.3. Linux-2.6.24.7

The Linux package contains the Linux kernel.

12.3.1. Installation of the kernel

A number of vulnerabilities have come to light after the stable kernel team stopped supporting 2.6.24. The following patch addresses them:

```
patch -Np1 -i ../linux-2.6.24.7-security_fixes-1.patch
```

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" and "tmpfs" 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.

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 Udev are not documented. The problem is that Udev 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 Udev:

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

Because of the complications with Udev and modules, we strongly recommend starting with a completely non-modular kernel configuration, especially if this is the first time using Udev.

Install the modules, if the kernel configuration uses them:

```
make modules_install
```

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

Issue the following command to install the kernel:

```
cp arch/x86_64/boot/bzImage /boot/clfskernel-2.6.24.7
```

`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-2.6.24.7
```

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-2.6.24.7
```

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-2.6.24.7` 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 (from the Linux-Headers package) and should *never* be replaced by the kernel headers.

12.3.2. Contents of Linux

Installed files: `config-[linux-version]`, `clfskernel-[linux-version]`, and `System.map-[linux-version]`

Short Descriptions

<code>config-[linux-version]</code>	Contains all the configuration selections for the kernel
<code>clfskernel-[linux-version]</code>	The engine of the Linux system. When turning on the computer, the kernel is the first part of the operating system that gets loaded. It detects and initializes all components of the computer's hardware, then makes these

components available as a tree of files to the software and turns a single CPU into a multitasking machine capable of running scores of programs seemingly at the same time.

`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

12.4. 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 computers using Lilo, which in the context of this book means x86_64 Pure64 systems. Information on “boot loading” for other architectures should be available in the usual resource-specific locations for those architectures.

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

If you have multiple systems on your machine using a different bootloader such as GRUB, you may prefer to use that instead - consult the appropriate documentation. The rest of this section assumes you are going to use Lilo.

Earlier, we compiled and installed the Lilo boot loader software in preparation for this step. The procedure involves writing a boot image to a specific location on the hard drive. We highly recommend using `mkrescue` to create a Lilo boot CD (using e.g. `dvdrecord` from `dvdtools`) as a backup (this requires loopback block device support in the kernel).

Normally, you interact with Lilo by using the cursor and `enter` keys to select from the available option(s), but sometimes it is necessary to add other boot options, such as e.g. `'init=/bin/bash'` to debug boot failures. The more your keyboard layout differs from the US qwerty layout, the harder it becomes to type boot options unless Lilo knows about your keyboard layout. So, we will create a key table for Lilo (`.klt`) file - at one point in the documentation these are referred to as `.klt` files, which may be a typo, but has been followed by some distros. The name, and location, are not important but it is conventional to put these in `/boot` with the name representing the key layout. For a British keyboard layout, the following command will achieve this:

```
keytab-lilo.pl uk >/boot/uk.klt
```

The argument to the command is the name of the keymap, or if necessary you can specify the full path to the keymap. Use whatever is appropriate for your keyboard.

When the x86 CLFS book used to include Lilo, it advised against running it from `chroot` in case the MBR became corrupted. Provided you have `/proc` mounted and have device special files for the disks, it seems to be safe to run recent versions of Lilo in `chroot`, although it is always possible that an updated bootloader, or defective configuration file, may render the system unbootable.

The next step is to create `/etc/lilo.conf`:

```
cat > /etc/lilo.conf << "EOF"
# Begin /etc/lilo.conf
# lilo.conf
#
# global options
boot=/dev/<bootdisk>
keytable=/boot/<keytable>
lba32
map=/boot/map
prompt

# set the name of the default image to boot
default=clfs

# define an image
image=/boot/clfskernel
    label=clfs
    root=/dev/<partition>
    read-only
# optionally add parameters to pass, e.g.
#   append="video=radeonfb:1024x768-16@70"

# repeat for any other kernel images

# optionally, add legacy operating systems
# see man lilo.conf for examples
EOF
```

Replace `<bootdisk>` with the name of the disk (or partition) on which the boot sector is to be written, e.g. `sda`. Replace `<keytable>` with the name of the keytable file you created, and `<partition>` with the name of the root partition for the new system.



Warning

The following command will overwrite the current boot loader. Do not run the command if this is not desired.

Run Lilo:

```
/sbin/lilo -v
```



Note

People who have been used to GRUB need to be aware that Lilo works differently - in particular, you cannot edit the available choices as you can in the **grub** shell, and Lilo records the block addresses of the kernels into the boot blocks each time `/sbin/lilo` is run. This means that when you compile a new kernel, you have to add it to `/etc/lilo.conf` and rerun `/sbin/lilo`. It also means that if you recompile an existing kernel and save it to the same name you still have to rerun `/sbin/lilo` in case it now occupies different blocks on the filesystem.

If you are running multiple systems on this box and using Lilo, it is a good idea to ensure that each system is running the same version of Lilo, otherwise an old version may not be able to overwrite the bootloader from a newer version. You will also need to ensure that the copies of `/etc/lilo.conf` on each system are kept synchronised.

Chapter 13. The End

13.1. 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 1.1.0 > /etc/clfs-release
```

13.2. Get Counted

Now that you have finished the book, do you want to be counted as a CLFS user? Head over to <http://www.linuxfromscratch.org/cgi-bin/lfscounter.cgi> and register as a CLFS user by entering your name and the first CLFS version you have used.

13.3. 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 the BLFS book while still in our current chroot environment, you can leave yourself in a much better position to continue on once you reboot into your new CLFS installation. Installing a text mode web browser, such as Lynx, you can easily view the BLFS book 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 Dhcpcd 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
umount ${CLFS}/dev/shm
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 1.1.0* 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.

13.4. 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:

- Freshmeat.net (<http://freshmeat.net/>)

Freshmeat 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/>.

- 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
ALFS	Automated Linux From Scratch
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
BLFS	Beyond Linux From Scratch
BSD	Berkeley Software Distribution
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
EVMS	Enterprise Volume Management System
ext2	second 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	Gibabytes
GCC	GNU Compiler Collection
GID	Group Identifier
GMT	Greenwich Mean Time
HTML	Hypertext Markup Language

IDE	Integrated Drive Electronics
IEEE	Institute of Electrical and Electronic Engineers
IO	Input/Output
IP	Internet Protocol
IPC	Inter-Process Communication
IRC	Internet Relay Chat
ISO	International Organization for Standardization
ISP	Internet Service Provider
KB	Kilobytes
LED	Light Emitting Diode
LFS	Linux From Scratch
LSB	Linux Standard Base
MB	Megabytes
MBR	Master Boot Record
MD5	Message Digest 5
NIC	Network Interface Card
NLS	Native Language Support
NNTP	Network News Transport Protocol
NPTL	Native POSIX Threading Library
OF	Open Firmware
OSS	Open Sound System
PCH	Pre-Compiled Headers
PCRE	Perl Compatible Regular Expression
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
SMP	Symmetric Multi-Processor
TLDP	The Linux Documentation Project

TFTP	Trivial File Transfer Protocol
TLS	Thread-Local Storage
UID	User Identifier
umask	user file-creation mask
USB	Universal Serial Bus
UTC	Coordinated Universal Time
UUID	Universally Unique Identifier
VC	Virtual Console
VGA	Video Graphics Array
VT	Virtual Terminal

Appendix B. Acknowledgments

We would like to thank the following people and organizations for their contributions to the Linux From Scratch Project.

- *Gerard Beekmans* <gerard@linuxfromscratch.org> – LFS Creator, LFS Project Leader
- *Matthew Burgess* <matthew@linuxfromscratch.org> – LFS Project Leader, LFS Technical Writer/Editor, LFS Release Manager
- *Archaic* <archaic@linuxfromscratch.org> – LFS Technical Writer/Editor, HLFS Project Leader, BLFS Editor, Hints and Patches Project Maintainer
- *Nathan Coulson* <nathan@linuxfromscratch.org> – LFS-Bootscripts Maintainer
- *Bruce Dubbs* <bdubbs@linuxfromscratch.org> – BLFS Project Leader
- *Manuel Canales Esparcia* <manuel@linuxfromscratch.org> – LFS/BLFS/HLFS XML and XSL Maintainer
- *Jim Gifford* <jim@linuxfromscratch.org> – LFS Technical Writer, Patches Project Leader
- *Jeremy Huntwork* <jhuntwork@linuxfromscratch.org> – LFS Technical Writer, LFS LiveCD Maintainer, ALFS Project Leader
- *Anderson Lizardo* <lizardo@linuxfromscratch.org> – Website Backend-Scripts Maintainer
- *Ryan Oliver* <ryan@linuxfromscratch.org> – LFS Toolchain Maintainer
- *James Robertson* <jwrober@linuxfromscratch.org> – Bugzilla Maintainer
- *Tushar Teredesai* <tushar@linuxfromscratch.org> – BLFS Book Editor, Hints and Patches Project Leader
- Countless other people on the various LFS and BLFS mailing lists who helped make this book possible by giving their suggestions, testing the book, and submitting bug reports, instructions, and their experiences with installing various packages.

Translators

- *Manuel Canales Esparcia* <macana@lfs-es.com> – Spanish LFS translation project
- *Johan Lenglet* <johan@linuxfromscratch.org> – French LFS translation project
- *Anderson Lizardo* <lizardo@linuxfromscratch.org> – Portuguese LFS translation project
- *Thomas Reitelbach* <tr@erdfunkstelle.de> – German LFS translation project

Mirror Maintainers

North American Mirrors

- *Scott Kveton* <scott@osuosl.org> – lfs.oregonstate.edu mirror
- *Mikhail Pastukhov* <miha@xuy.biz> – lfs.130th.net mirror
- *William Astle* <lost@l-w.net> – ca.linuxfromscratch.org mirror
- *Jeremy Polen* <jpolen@rackspace.com> – us2.linuxfromscratch.org mirror
- *Tim Jackson* <tim@idge.net> – linuxfromscratch.idge.net mirror
- *Jeremy Utley* <jeremy@linux-phreak.net> – lfs.linux-phreak.net mirror

South American Mirrors

- *Andres Meggiotto* <sysop@mesi.com.ar> – lfs.mesi.com.ar mirror
- *Manuel Canales Esparcia* <manuel@linuxfromscratch.org> – lfsmirror.lfs-es.info mirror
- *Eduardo B. Fonseca* <ebf@aedsolucoes.com.br> – br.linuxfromscratch.org mirror

European Mirrors

- *Barna Koczka* <barna@siker.hu> – hu.linuxfromscratch.org mirror
- *UK Mirror Service* – linuxfromscratch.mirror.ac.uk mirror
- *Martin Voss* <Martin.Voss@ada.de> – lfs.linux-matrix.net mirror
- *Guido Passet* <guido@primerelay.net> – nl.linuxfromscratch.org mirror
- *Bastiaan Jacques* <baafie@planet.nl> – lfs.pagefault.net mirror
- *Roel Neefs* <lfs-mirror@linuxfromscratch.rave.org> – linuxfromscratch.rave.org mirror
- *Justin Knierim* <justin@jrknierim.de> – www.lfs-matrix.de mirror
- *Stephan Brendel* <stevie@stevie20.de> – lfs.netservice-neuss.de mirror
- *Antonin Sprinzl* <Antonin.Sprinzl@tuwien.ac.at> – at.linuxfromscratch.org mirror
- *Fredrik Danerklint* <fredan-lfs@fredan.org> – se.linuxfromscratch.org mirror
- *Parisian sysadmins* <archive@doc.cs.univ-paris8.fr> – www2.fr.linuxfromscratch.org mirror
- *Alexander Velin* <velin@zadnik.org> – bg.linuxfromscratch.org mirror
- *Dirk Webster* <dirk@securewebservices.co.uk> – lfs.securewebservices.co.uk mirror
- *Thomas Skyt* <thomas@sofagang.dk> – dk.linuxfromscratch.org mirror
- *Simon Nicoll* <sime@dot-sime.com> – uk.linuxfromscratch.org mirror

Asian Mirrors

- *Pui Yong* <pyng@spam.averse.net> – sg.linuxfromscratch.org mirror
- *Stuart Harris* <stuart@althalus.me.uk> – lfs.mirror.intermedia.com.sg mirror

Australian Mirrors

- *Jason Andrade* <jason@dstc.edu.au> – au.linuxfromscratch.org mirror

Former Project Team Members

- *Christine Barczak* <theladyskye@linuxfromscratch.org> – LFS Book Editor
- Timothy Bauscher
- Robert Briggs
- Ian Chilton
- *Jeroen Coumans* <jeroen@linuxfromscratch.org> – Website Developer, FAQ Maintainer
- Alex Groenewoud – LFS Technical Writer
- Marc Heerdink

- Mark Hymers
- Seth W. Klein – FAQ maintainer
- *Nicholas Leippe* <nicholas@linuxfromscratch.org> – Wiki Maintainer
- Simon Perreault
- *Scot Mc Pherson* <scot@linuxfromscratch.org> – LFS NNTP Gateway Maintainer
- *Alexander Patrakov* <semzx@newmail.ru> – LFS Technical Writer
- *Greg Schafer* <gschafer@zip.com.au> – LFS Technical Writer
- Jesse Tie-Ten-Quee – LFS Technical Writer
- *Jeremy Utley* <jeremy@linuxfromscratch.org> – LFS Technical Writer, Bugzilla Maintainer, LFS-Bootscripts Maintainer
- *Zack Winkles* <zwinkles@gmail.com> – LFS Technical Writer

A very special thank you to our donators

- *Dean Benson* <dean@vipersoft.co.uk> for several monetary contributions
- *Hagen Herrschaft* <hrx@hrxnet.de> for donating a 2.2 GHz P4 system, now running under the name of Lorien
- *VA Software* who, on behalf of *Linux.com*, donated a VA Linux 420 (former StartX SP2) workstation
- Mark Stone for donating Belgarath, the linuxfromscratch.org server

Appendix C. 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, Grep, M4, Make, Perl, Sed and Texinfo
Test suite depends on: Automake, Binutils, Diffutils, Findutils, Gawk, GCC and Libtool
Must be installed before: Automake

Automake

Installation depends on: Autoconf, Bash, Coreutils, Gettext, Grep, M4, Make, Perl, Sed and Texinfo
Test suite depends on: Binutils, Bison, Bzip2, DejaGNU, Diffutils, Expect, Findutils, Flex, GCC, Gettext, Gzip, Libtool and Tar. 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, Gawk, GCC, Glibc, Grep, Make, Ncurses, Patch, Readline, Sed and Texinfo
Test suite depends on: None
Must be installed before: None

Binutils

Installation depends on: Bash, Binutils, Coreutils, Diffutils, File, Gawk, GCC, Gettext, Glibc, Grep, Make, Perl, Sed and Texinfo
Test suite depends on: DejaGNU and Expect
Must be installed before: None

Bison

Installation depends on: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, 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, Diffutils, GCC, Glibc, Make
Test suite depends on: None
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

Coreutils

Installation depends on: Bash, Binutils, Coreutils, Gawk, GCC, Gettext, Glibc, Grep, Make, Patch, Perl, Sed and Texinfo
Test suite depends on: Diffutils, E2fsprogs
Must be installed before: Bash, Diffutils, Findutils, Man and Udev

DejaGNU

Installation depends on: Bash, Coreutils, Diffutils, GCC, Grep, Make and Sed
Test suite depends on: None
Must be installed before: None

Diffutils

Installation depends on: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, Make, Patch, Sed and Texinfo
Test suite depends on: No testsuite available
Must be installed before: None

Expect

Installation depends on: Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Grep, Make, Patch, Sed and Tcl
Test suite depends on: None
Must be installed before: None

E2fsprogs

Installation depends on: Bash, Binutils, Coreutils, Gawk, GCC, Gettext, Glibc, Grep, Gzip, Make, Sed and Texinfo
Test suite depends on: Bzip2 and Diffutils
Must be installed before: Util-linux-ng

File

Installation depends on: Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Make, Sed and Zlib
Test suite depends on: No testsuite available
Must be installed before: None

Findutils

Installation depends on: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, Make, Sed and Texinfo
Test suite depends on: DejaGNU, Diffutils, Expect and Gawk
Must be installed before: None

Flex

Installation depends on: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, 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, GCC, Gettext, Glibc, Grep, Make, Patch, Sed and Texinfo
Test suite depends on: Diffutils
Must be installed before: None

Gcc

Installation depends on: Bash, Binutils, Coreutils, Diffutils, Findutils, Gawk, GCC, Gettext, Glibc, Grep, Make, Patch, Perl, Sed, Tar and Texinfo
Test suite depends on: DejaGNU and Expect
Must be installed before: None

Gettext

Installation depends on: Bash, Binutils, Coreutils, Gawk, GCC, Glibc, Grep, Make, Sed and Texinfo
Test suite depends on: Diffutils, Perl and Tcl
Must be installed before: Automake

Glibc

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

Grep

Installation depends on: Bash, Binutils, Coreutils, Diffutils, GCC, Gettext, Glibc, Grep, Make, Sed and Texinfo
Test suite depends on: Gawk
Must be installed before: Man

Groff

Installation depends on: Bash, Binutils, Bison, Coreutils, Gawk, GCC, Glibc, Grep, Make, Sed and Texinfo
Test suite depends on: No testsuite available
Must be installed before: Man and Perl

Gzip

Installation depends on: Bash, Binutils, Coreutils, GCC, Glibc, 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

Inetutils

Installation depends on: Bash, Binutils, Coreutils, GCC, Glibc, Grep, Make, Ncurses, Patch, Sed and Texinfo
Test suite depends on: No testsuite available
Must be installed before: Tar

IProute2

Installation depends on: Bash, Binutils, Bison, Coreutils, Flex, GCC, Glibc, Make and Linux-Headers
Test suite depends on: No testsuite available
Must be installed before: None

Kbd

Installation depends on: Bash, Binutils, Bison, Coreutils, Flex, GCC, Gettext, Glibc, Gzip, Make, and Sed
Test suite depends on: No testsuite available
Must be installed before: None

Less

Installation depends on: Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Grep, Make, Ncurses and Sed
Test suite depends on: No testsuite available
Must be installed before: None

Libtool

Installation depends on: Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Make, Sed and Texinfo
Test suite depends on: Findutils
Must be installed before: None

Linux-Headers

Installation depends on: Binutils, Coreutils, Findutils, GCC, Grep, Make and Sed
Test suite depends on: No testsuite available
Must be installed before: None

Linux Kernel

Installation depends on: Bash, Binutils, Coreutils, Diffutils, Findutils, GCC, Glibc, Grep, Gzip, Make, Mktemp, Module-Init-Tools, Ncurses and Sed
Test suite depends on: No testsuite available
Must be installed before: None

M4

Installation depends on: Bash, Binutils, Coreutils, GCC, Glibc, Grep, Make, Sed and Texinfo
Test suite depends on: Diffutils
Must be installed before: Autoconf and Bison

Make

Installation depends on: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, 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, Flex, Gawk, GCC, Gettext, Glibc, Grep, Groff, Gzip, Less, Make and Sed
Test suite depends on: No testsuite available
Must be installed before: None

Mktemp

Installation depends on: Bash, Binutils, Coreutils, GCC, Glibc, Grep, Patch and Sed
Test suite depends on: No testsuite available
Must be installed before: None

Module-Init-Tools

Installation depends on: Bash, Binutils, Coreutils, Findutils, GCC, Glibc, Grep, Make, Sed and Zlib
Test suite depends on: Diffutils, File, Gawk, Gzip, and Mktemp
Must be installed before: None

Ncurses

Installation depends on: Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, 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-ng and Vim

Patch

Installation depends on: Bash, Binutils, Coreutils, GCC, Glibc, Grep, Make and Sed
Test suite depends on: No testsuite available
Must be installed before: None

Perl

Installation depends on: Bash, Binutils, Coreutils, Gawk, GCC, Glibc, Grep, Groff, Make and Sed
Test suite depends on: Iana-Etc and Procps
Must be installed before: Autoconf

Procps

Installation depends on: Bash, Binutils, Coreutils, GCC, Glibc, Make and Ncurses
Test suite depends on: No testsuite available
Must be installed before: None

Psmisc

Installation depends on: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, Make, Ncurses and Sed
Test suite depends on: No testsuite available
Must be installed before: None

Readline

Installation depends on: Bash, Binutils, Coreutils, GCC, Glibc, Grep, Make, Ncurses, Patch, Sed and Texinfo
Test suite depends on: No testsuite available
Must be installed before: Bash

Sed

Installation depends on: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, Make, Sed and Texinfo
Test suite depends on: Diffutils and Gawk
Must be installed before: E2fsprogs, File, Libtool and Shadow

Shadow

Installation depends on: Bash, Binutils, Coreutils, Diffutils, Findutils, Gawk, GCC, Gettext, Glibc, Grep, Make and Sed
Test suite depends on: No testsuite available
Must be installed before: None

Sysklogd

Installation depends on: Binutils, Coreutils, GCC, Glibc and Make
Test suite depends on: No testsuite available
Must be installed before: None

Sysvinit

Installation depends on: Binutils, Coreutils, GCC, Glibc, Make and Sed
Test suite depends on: No testsuite available
Must be installed before: None

Tar

Installation depends on: Bash, Binutils, Bison, Coreutils, GCC, Gettext, Glibc, Grep, Inetutils, 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, GCC, Glibc, Grep, Make and Sed
Test suite depends on: None
Must be installed before: None

Texinfo

Installation depends on: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, Make, Ncurses, Patch and Sed
Test suite depends on: Diffutils
Must be installed before: None

Tree

Installation depends on: Coreutils, GCC and Make
Test suite depends on: None
Must be installed before: None

Udev

Installation depends on: Binutils, Coreutils, GCC, Glibc and Make
Test suite depends on: Findutils, Perl and Sed
Must be installed before: None

Util-linux-ng

Installation depends on: Bash, Binutils, Coreutils, E2fprogs, GCC, Gettext, Glibc, Grep, Make, Ncurses, Sed, Texinfo and Zlib
Test suite depends on: No testsuite available
Must be installed before: None

Vim

Installation depends on: Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Grep, Make, Ncurses and Sed
Test suite depends on: None
Must be installed before: None

Zlib

Installation depends on: Bash, Binutils, Coreutils, GCC, Glibc, Grep, Make and Sed
Test suite depends on: None
Must be installed before: File, Module-Init-Tools and Util-linux-ng

Appendix D. x86 Dependencies

This page contains dependency information for packages specific to x86 Pure64.

Bin86

Installation depends on: Bash, Binutils, Coreutils, GCC, Glibc, Make and Patch

Test suite depends on: None

Must be installed before: Lilo

Lilo

Installation depends on: Bash, Bin86, Binutils, Coreutils, GCC, Glibc, Make and Sed

Test suite depends on: None

Must be installed before: None

Appendix E. 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: 154
- Automake: 155
- Bash: 157
 - temporary system: 55
- Bin86: 204
 - boot: 87
- Binutils: 124
 - cross tools: 41
 - temporary system: 51
- Bison: 140
- Bootscripts: 208
 - boot: 91
 - usage: 209
- Bzip2: 159
 - temporary system: 57
- Coreutils: 133
 - temporary system: 58
- DejaGNU: 108
- Diffutils: 161
 - temporary system: 59
- E2fsprogs: 130
 - boot: 75
 - chroot: 96
- Expect: 107
- File: 40, 62, 162
- Findutils: 163
 - temporary system: 60
- Flex: 146
- Gawk: 165
 - temporary system: 63
- GCC: 127
 - cross tools, final: 47
 - cross tools, static: 43
 - temporary system: 52
- Gettext: 166
 - temporary system: 64
- Glibc: 117
 - cross tools: 45
- Grep: 168
 - temporary system: 65
- Groff: 169
- Gzip: 172
 - temporary system: 66
- Iana-Etc: 138
- Inetutils: 174
- IPRoute2: 147
- Kbd: 176
- Less: 178
- Libtool: 145
- Lilo: 205
 - boot: 88
 - configuring: 92, 227
- Linux: 224
 - boot: 84
- Linux-Headers: 115
 - cross tools: 39
- M4: 139
- Make: 179
 - temporary system: 67
- Man: 180
- Man-pages: 116
- Mktemp: 182
- Module-Init-Tools: 183
 - boot: 78
- Ncurses: 141
 - temporary system: 54
- Patch: 185
 - temporary system: 68
- Perl: 149
 - temporary tools: 114
- Procps: 143
- Psmisc: 186
- Readline: 152
- Sed: 129
 - temporary system: 69
- Shadow: 187
 - configuring: 188
- Sysklogd: 190
 - configuring: 190
- Sysvinit: 191
 - boot: 76
 - boot, configuring: 77
 - configuring: 192
- Tar: 194
 - temporary system: 70
- Tcl: 106
- Texinfo: 195
 - temporary system: 71
- Tree: 109
- Udev: 196

boot: 80
 usage: 210
 Util-linux-ng: 198
 boot: 79
 chroot: 97
 Vim: 201
 Zlib: 153
 boot: 61

Programs

a2p: 149, 150
 acinstall: 155, 155
 aclocal: 155, 155
 aclocal-1.10.1: 155, 155
 addftinfo: 169, 169
 addpart: 198, 198
 addr2line: 124, 125
 afmtodit: 169, 169
 agetty: 198, 198
 apropos: 180, 181
 ar: 124, 125
 as: 124, 125
 as86: 204, 204
 ata_id: 196, 197
 autoconf: 154, 154
 autoheader: 154, 154
 autom4te: 154, 154
 automake: 155, 155
 automake-1.10.1: 155, 155
 autopoint: 166, 166
 autoreconf: 154, 154
 autoscan: 154, 154
 autoupdate: 154, 154
 awk: 165, 165
 badblocks: 130, 131
 base64: 133, 134
 basename: 133, 134
 bash: 157, 157
 bashbug: 157, 158
 bigram: 163, 163
 bison: 140, 140
 blkid: 130, 131
 blockdev: 198, 198
 bootlogd: 191, 192
 bunzip2: 159, 159
 bzcac: 159, 159
 bzcmp: 159, 159
 bzdiff: 159, 159
 bzegrep: 159, 159
 bzfgrep: 159, 159
 bzgrep: 159, 159
 bzip2: 159, 160
 bzip2recover: 159, 160
 bzless: 159, 160
 bzmoe: 159, 160
 c++: 127, 128
 c++filt: 124, 125
 c2ph: 149, 150
 cal: 198, 198
 captinfo: 141, 141
 cat: 133, 134
 catchsegv: 117, 121
 cc: 127, 128
 cdrom_id: 196, 197
 cfdisk: 198, 199
 chage: 187, 189
 chatr: 130, 131
 chfn: 187, 189
 chgpasswd: 187, 189
 chgrp: 133, 134
 chkdupexe: 198, 199
 chmod: 133, 134
 chown: 133, 134
 chpasswd: 187, 189
 chroot: 133, 134
 chsh: 187, 189
 chvt: 176, 176
 cksum: 133, 134
 clear: 141, 141
 clfskernel-[linux-version]: 224, 225
 cmp: 161, 161
 code: 163, 163
 col: 198, 199
 colcrt: 198, 199
 colrm: 198, 199
 column: 198, 199
 comm: 133, 134
 compile: 155, 155
 compile_et: 130, 131
 config.guess: 155, 155
 config.sub: 155, 155
 cp: 133, 134
 cpan: 149, 150
 cpp: 127, 128

create_floppy_devices: 196, 197
 csplit: 133, 134
 ctrlaltdel: 198, 199
 ctstat: 147, 147
 cut: 133, 134
 cytune: 198, 199
 dasd_id: 196, 197
 date: 133, 134
 dd: 133, 134
 ddate: 198, 199
 deallocvt: 176, 176
 debugfs: 130, 131
 delpart: 198, 199
 depcomp: 155, 155
 depmod: 183, 183
 df: 133, 134
 diag1.img: 205, 205
 diff: 161, 161
 diff3: 161, 161
 dir: 133, 135
 dircolors: 133, 135
 dirname: 133, 135
 dmesg: 198, 199
 dprofpp: 149, 150
 du: 133, 135
 dumpe2fs: 130, 131
 dumpkeys: 176, 176
 e2fsck: 130, 131
 e2image: 130, 131
 e2label: 130, 131
 echo: 133, 135
 edd_id: 196, 197
 efm_filter.pl: 201, 202
 efm_perl.pl: 201, 202
 egrep: 168, 168
 elisp-comp: 155, 155
 enc2xs: 149, 150
 env: 133, 135
 envsubst: 166, 166
 eqn: 169, 169
 eqn2graph: 169, 169
 ex: 201, 202
 expand: 133, 135
 expect: 107, 107
 expiry: 187, 189
 expr: 133, 135
 factor: 133, 135
 faillog: 187, 189
 false: 133, 135
 fdformat: 198, 199
 fdisk: 198, 199
 fgconsole: 176, 176
 fgrep: 168, 168
 file: 162, 162
 filefrag: 130, 131
 find: 163, 163
 find2perl: 149, 150
 findfs: 130, 131
 firmware.sh: 196, 197
 flex: 146, 146
 flock: 198, 199
 fmt: 133, 135
 fold: 133, 135
 frcode: 163, 163
 free: 143, 143
 fsck: 130, 131
 fsck.cramfs: 198, 199
 fsck.ext2: 130, 131
 fsck.ext3: 130, 131
 fsck.minix: 198, 199
 ftp: 174, 175
 fuser: 186, 186
 g++: 127, 128
 gawk: 165, 165
 gawk-3.1.6: 165, 165
 gcc: 127, 128
 gccbug: 127, 128
 gcov: 127, 128
 gdiffmk: 169, 169
 gencat: 117, 121
 generate-modprobe.conf: 183, 184
 geqn: 169, 169
 getconf: 117, 121
 getent: 117, 121
 getkeycodes: 176, 176
 getopt: 198, 199
 gettext: 166, 166
 gettext.sh: 166, 166
 gettextize: 166, 166
 gpasswd: 187, 189
 gprof: 124, 126
 grap2graph: 169, 169
 grcat: 165, 165
 grep: 168, 168

grn: 169, 169
 grodvi: 169, 169
 groff: 169, 170
 groffer: 169, 170
 grog: 169, 170
 grolbp: 169, 170
 grolj4: 169, 170
 grops: 169, 170
 grotty: 169, 170
 groupadd: 187, 189
 groupdel: 187, 189
 groupmod: 187, 189
 groups: 133, 135
 grpck: 187, 189
 grpconv: 187, 189
 grpunconv: 187, 189
 gtbl: 169, 170
 gunzip: 172, 172
 gzexe: 172, 172
 gzip: 172, 172
 h2ph: 149, 150
 h2xs: 149, 150
 halt: 191, 193
 head: 133, 135
 hexdump: 198, 199
 hostid: 133, 135
 hostname: 133, 135
 hostname: 166, 166
 hpftodit: 169, 170
 hwclock: 198, 199
 iconv: 117, 121
 iconvconfig: 117, 121
 id: 133, 135
 ifcfg: 147, 147
 ifconfig: 174, 175
 ifnames: 154, 154
 ifstat: 147, 147
 igawk: 165, 165
 indxbib: 169, 170
 info: 195, 195
 infocmp: 141, 142
 infokey: 195, 195
 infotocap: 141, 142
 init: 191, 193
 insmod: 183, 184
 insmod.static: 183, 184
 install: 133, 135
 install-info: 195, 195
 install-sh: 155, 155
 instmodsh: 149, 150
 ionice: 198, 199
 ip: 147, 147
 ipcrm: 198, 199
 ipcs: 198, 199
 isosize: 198, 199
 join: 133, 135
 kbdrate: 176, 176
 kbd_mode: 176, 176
 keytab-lilo.pl: 205, 205
 kill: 143, 143
 killall: 186, 186
 killall5: 191, 193
 klogd: 190, 190
 last: 191, 193
 lastb: 191, 193
 lastlog: 187, 189
 ld: 124, 126
 ld86: 204, 204
 ldconfig: 117, 121
 ldd: 117, 121
 lddlibc4: 117, 121
 less: 178, 178
 less.sh: 201, 202
 lessecho: 178, 178
 lesskey: 178, 178
 lex: 146, 146
 libnetcfg: 149, 150
 libtool: 145, 145
 libtoolize: 145, 145
 libvolume_id: 196, 197
 lilo: 205, 205
 line: 198, 199
 link: 133, 135
 lkbib: 169, 170
 ln: 133, 135
 lnstat: 147, 148
 loadkeys: 176, 176
 loadunimap: 176, 176
 locale: 117, 121
 localedef: 117, 121
 locate: 163, 163
 logger: 198, 199
 login: 187, 189
 logname: 133, 135

logout: 187, 189
logsave: 130, 131
look: 198, 199
lookbib: 169, 170
losetup: 198, 199
ls: 133, 135
lsattr: 130, 131
lsmod: 183, 184
m4: 139, 139
make: 179, 179
makeinfo: 195, 195
makewhatis: 180, 181
man: 180, 181
man2dvi: 180, 181
man2html: 180, 181
mapscrn: 176, 177
mcookie: 198, 199
md5sum: 133, 135
mdate-sh: 155, 155
mesg: 191, 193
missing: 155, 155
mkdir: 133, 135
mke2fs: 130, 131
mkfifo: 133, 135
mkfs: 198, 199
mkfs.bfs: 198, 199
mkfs.cramfs: 198, 199
mkfs.ext2: 130, 131
mkfs.ext3: 130, 131
mkfs.minix: 198, 199
mkinstalldirs: 155, 155
mklost+found: 130, 131
mknod: 133, 135
mkrescue: 205, 205
mkswap: 198, 199
mktemp: 182, 182
mk_cmds: 130, 131
mmroff: 169, 170
modinfo: 183, 184
modprobe: 183, 184
more: 198, 199
mount: 198, 199
mountpoint: 191, 193
msgattrib: 166, 166
msgcat: 166, 166
msgcmp: 166, 166
msgcomm: 166, 166
msgconv: 166, 166
msgen: 166, 166
msgexec: 166, 166
msgfilter: 166, 166
msgfmt: 166, 167
msggrep: 166, 167
msginit: 166, 167
msgmerge: 166, 167
msgunfmt: 166, 167
msguniq: 166, 167
mtrace: 117, 121
mv: 133, 135
mve.awk: 201, 202
namei: 198, 200
ncurses5-config: 141, 142
neqn: 169, 170
newgrp: 187, 189
newusers: 187, 189
ngettext: 166, 167
nice: 133, 135
nl: 133, 135
nm: 124, 126
nm86: 204, 204
nohup: 133, 135
nologin: 187, 189
nroff: 169, 170
nscd: 117, 121
nstat: 147, 148
objcopy: 124, 126
objdump: 124, 126
objdump86: 204, 204
od: 133, 135
oldfuser: 186, 186
openvt: 176, 177
partx: 198, 200
passwd: 187, 189
paste: 133, 135
patch: 185, 185
pathchk: 133, 135
path_id: 196, 197
pcprofiledump: 117, 121
pdfroff: 169, 170
perl: 149, 150
perl5.8.8: 149, 150
perlbug: 149, 150
perlcc: 149, 150
perldoc: 149, 150

perlivp: 149, 150
pfbtops: 169, 170
pg: 198, 200
pgawk: 165, 165
pgawk-3.1.6: 165, 165
pgrep: 143, 143
pic: 169, 170
pic2graph: 169, 170
piconv: 149, 150
pidof: 191, 193
ping: 174, 175
ping6: 174, 175
pinky: 133, 136
pivot_root: 198, 200
pkill: 143, 143
pl2pm: 149, 150
pltags.pl: 201, 203
pmap: 143, 143
pod2html: 149, 151
pod2latex: 149, 151
pod2man: 149, 151
pod2text: 149, 151
pod2usage: 149, 151
podchecker: 149, 151
podselect: 149, 151
post-grohtml: 169, 170
poweroff: 191, 193
pr: 133, 136
pre-grohtml: 169, 170
printenv: 133, 136
printf: 133, 136
prove: 149, 151
ps: 143, 143
psed: 149, 151
psfaddtable: 176, 177
psfgettable: 176, 177
psfstriutable: 176, 177
psfxtable: 176, 177
pstree: 186, 186
pstree.x11: 186, 186
pstruct: 149, 151
ptx: 133, 136
pt_chown: 117, 121
pwcats: 165, 165
pwck: 187, 189
pwconv: 187, 189
pwd: 133, 136
pwdx: 143, 143
pwunconv: 187, 189
py-compile: 155, 156
ranlib: 124, 126
rcp: 174, 175
readelf: 124, 126
readlink: 133, 136
readprofile: 198, 200
reboot: 191, 193
recode-sr-latin: 166, 167
ref: 201, 203
refer: 169, 170
rename: 198, 200
renice: 198, 200
reset: 141, 142
resize2fs: 130, 131
resizecons: 176, 177
rev: 198, 200
rlogin: 174, 175
rm: 133, 136
rmdir: 133, 136
rmmod: 183, 184
rmt: 194, 194
routef: 147, 148
routel: 147, 148
rpcgen: 117, 121
rpcinfo: 117, 121
rsh: 174, 175
rtacct: 147, 148
rtcwake: 198, 200
rtmon: 147, 148
rtpr: 147, 148
rtstat: 147, 148
runlevel: 191, 193
runttest: 108, 108
rview: 201, 203
rvim: 201, 203
s2p: 149, 151
script: 198, 200
scriptreplay: 198, 200
scsi_id: 196, 197
sdiff: 161, 161
sed: 129, 129
seq: 133, 136
setarch: 198, 200
setfont: 176, 177
setkeycodes: 176, 177

settled: 176, 177
 setmetamode: 176, 177
 setsid: 198, 200
 setterm: 198, 200
 sfdisk: 198, 200
 sg: 187, 189
 sh: 157, 158
 sha1sum: 133, 136
 sha224sum: 133, 136
 sha256sum: 133, 136
 sha384sum: 133, 136
 sha512sum: 133, 136
 showconsolefont: 176, 177
 showkey: 176, 177
 shred: 133, 136
 shtags.pl: 201, 203
 shuf: 133, 136
 shutdown: 191, 193
 size: 124, 126
 size86: 204, 204
 skill: 143, 143
 slabtop: 143, 143
 sleep: 133, 136
 sln: 117, 121
 snice: 143, 143
 soelim: 169, 170
 sort: 133, 136
 splain: 149, 151
 split: 133, 136
 sproff: 117, 122
 ss: 147, 148
 stat: 133, 136
 strings: 124, 126
 strip: 124, 126
 stty: 133, 136
 su: 187, 189
 sulogin: 191, 193
 sum: 133, 136
 swapoff: 198, 200
 swapon: 198, 200
 symlink-tree: 155, 156
 sync: 133, 136
 sysctl: 143, 143
 syslogd: 190, 190
 tac: 133, 136
 tack: 141, 142
 tail: 133, 136
 tailf: 198, 200
 talk: 174, 175
 tar: 194, 194
 taskset: 198, 200
 tbl: 169, 170
 tc: 147, 148
 tcsh: 106, 106
 tcsh8.4: 106, 106
 tcltags: 201, 203
 tee: 133, 136
 telinit: 191, 193
 telnet: 174, 175
 tempfile: 182, 182
 test: 133, 136
 texi2dvi: 195, 195
 texi2pdf: 195, 195
 texindex: 195, 195
 tfmtodit: 169, 170
 tftp: 174, 175
 tic: 141, 142
 tload: 143, 143
 toe: 141, 142
 top: 143, 143
 touch: 133, 136
 tput: 141, 142
 tr: 133, 136
 tree: 109, 109
 troff: 169, 170
 true: 133, 136
 tset: 141, 142
 tsort: 133, 136
 tty: 133, 136
 tune2fs: 130, 131
 tunelp: 198, 200
 tzselect: 117, 122
 udevadm trigger: 196, 197
 udevadm: 196, 196
 udevadm control: 196, 197
 udevadm monitor: 196, 197
 udevadm test: 196, 197
 udevd: 196, 197
 udevinfo: 196, 197
 udevsettle: 196, 197
 ul: 198, 200
 umount: 198, 200
 uname: 133, 136
 uncompress: 172, 172

unexpand: 133, 137
 unicode_start: 176, 177
 unicode_stop: 176, 177
 uniq: 133, 137
 unlink: 133, 137
 updatedb: 163, 163
 uptime: 143, 143
 usb_id: 196, 197
 useradd: 187, 189
 userdel: 187, 189
 usermod: 187, 189
 users: 133, 137
 utmpdump: 191, 193
 uuid: 130, 131
 uuidgen: 130, 131
 vdir: 133, 137
 vi: 201, 203
 view: 201, 203
 vigr: 187, 189
 vim: 201, 203
 vim132: 201, 203
 vim2html.pl: 201, 203
 vimdiff: 201, 203
 vim: 201, 203
 vimspell.sh: 201, 203
 vimtutor: 201, 203
 vipw: 187, 189
 vmstat: 143, 143
 vol_id: 196, 197
 w: 143, 143
 wall: 191, 193
 watch: 143, 143
 wc: 133, 137
 whatis: 180, 181
 whereis: 198, 200
 who: 133, 137
 whoami: 133, 137
 write: 198, 200
 xargs: 163, 164
 xgettext: 166, 167
 xsubpp: 149, 151
 xtrace: 117, 122
 xxd: 201, 203
 yacc: 140, 140
 yes: 133, 137
 ylwrap: 155, 156
 zcat: 172, 172

zcmp: 172, 172
 zdiff: 172, 172
 zdump: 117, 122
 zegrep: 172, 172
 zfgrep: 172, 172
 zforce: 172, 172
 zgrep: 172, 172
 zic: 117, 122
 zless: 172, 172
 zmore: 172, 173
 znew: 172, 173
 zsoelim: 169, 171

Libraries

ld.so: 117, 122
 libanl: 117, 122
 libasprintf: 166, 167
 libbfd: 124, 126
 libblkid: 130, 131
 libBrokenLocale: 117, 122
 libbsd-compat: 117, 122
 libbz2*: 159, 160
 libc: 117, 122
 libcom_err: 130, 131
 libcrypt: 117, 122
 libcurses: 141, 142
 libdl: 117, 122
 libe2p: 130, 131
 libexpect-5.43: 107, 107
 libext2fs: 130, 131
 libfl.a: 146, 146, 146, 146
 libform: 141, 142
 libg: 117, 122
 libgcc*: 127, 128
 libgettextlib: 166, 167
 libgettextpo: 166, 167
 libgettextsrc: 166, 167
 libhistory: 152, 152
 libiberty: 124, 126
 libieee: 117, 122
 libltdl: 145, 145
 libm: 117, 122
 libmagic: 162, 162
 libmcheck: 117, 122
 libmemusage: 117, 122
 libmenu: 141, 142
 libmudflap*: 127, 128

libncurses: 141, 142
 libnsl: 117, 122
 libnss: 117, 122
 libopcodes: 124, 126
 libpanel: 141, 142
 libpcprofile: 117, 122
 libproc: 143, 144
 libpthread: 117, 122
 libreadline: 152, 152
 libresolv: 117, 122
 librpcsvc: 117, 122
 librt: 117, 122
 libSegFault: 117, 122
 libss: 130, 132
 libstdc++: 127, 128
 libsupc++: 127, 128
 libtcl8.4.so: 106, 106
 libthread_db: 117, 122
 libutil: 117, 122
 libuuid: 130, 132
 liby.a: 140, 140
 libz: 153, 153

Scripts

checkfs: 208, 208
 cleanfs: 208, 208
 console: 208, 208
 configuring: 214
 functions: 208, 208
 halt: 208, 208
 ifdown: 208, 208
 ifup: 208, 208
 localnet: 208, 208
 /etc/hosts: 218
 configuring: 218
 mountfs: 208, 208
 mountkernfs: 208, 208
 network: 208, 208
 /etc/hosts: 218
 configuring: 221
 rc: 208, 208
 reboot: 208, 208
 sendsignals: 208, 208
 setclock: 208, 208
 configuring: 213
 static: 208, 208
 swap: 208, 208

syslogd: 208, 208
 configuring: 215
 template: 208, 208

Others

/boot/config-[linux-version]: 224, 225
 /boot/System.map-[linux-version]: 224, 226
 /dev/*: 92, 103
 /etc/clfs-release: 230
 /etc/fstab: 89, 223
 /etc/group: 81, 101
 /etc/hosts: 218
 /etc/inittab: 77, 192
 /etc/inputrc: 215
 /etc/ld.so.conf: 120
 /etc/localtime: 119
 /etc/login.defs: 188
 /etc/nsswitch.conf: 119
 /etc/passwd: 81, 101
 /etc/profile: 216
 /etc/protocols: 138
 /etc/resolv.conf: 222
 /etc/services: 138
 /etc/syslog.conf: 190
 /etc/udev: 196, 197
 /etc/vimrc: 202
 /lib/udev: 196, 197
 /usr/include/{asm,linux}/*.h: 115, 115
 /var/log/btmp: 81, 101
 /var/log/lastlog: 81, 101
 /var/log/wtmp: 81, 101
 /var/run/utmp: 81, 101
 man pages: 116, 116