Cross-Compiled Linux From Scratch

Version 3.0.0-SYSVINIT-PowerPC

Cross-Compiled Linux From Scratch: Version 3.0.0-SYSVINIT-PowerPC

Table of Contents

Preface

Foreword

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

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

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

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

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

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

Audience

There are many reasons why somebody would want to read this book. The principal reason is to install a Linux system from the source code. A question many people raise is, “why go through all the hassle of manually building a Linux system from scratch when you can just download and install an existing one?” That is a good question and is the impetus for this section of the book.

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

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

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

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

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

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

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

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

Prerequisites

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

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

Before building a CLFS system, we recommend reading the following HOWTOs:

- Software-Building-HOWTO

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

This is a comprehensive guide to building and installing “generic” Unix software distributions under Linux.

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

  This guide covers the usage of assorted Linux software.

- The Essential Pre-Reading Hint
  http://hints.cross-lfs.org/index.php/Essential_Prereading

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

Host System Requirements

You should be able to build a CLFS system from just about any Unix-type operating system. Your host system should have the following software with the minimum versions indicated. Also note that many distributions will place software headers into separate packages, often in the form of “[package-name]-devel” or “[package-name]-dev”. Be sure to install those if your distribution provides them.

- **Bash-2.05a**
- **Binutils-2.12** (Versions greater than 2.24 are not recommended as they have not been tested)
- **Bison-1.875**
- **Bzip2-1.0.2**
- **Coreutils-5.0**
- **Diffutils-2.8**
- **Findutils-4.1.20**
- **Gawk-3.1.5**
- **GCC-4.1.2** and the C++ compiler, **g++** (Versions greater than 4.8.3 are not recommended as they have not been tested)
- **Glibc-2.2.5** (Versions greater than 2.19 are not recommended as they have not been tested)
- **Grep-2.5**
- **Gzip-1.2.4**
- **Make-3.80**
- **Ncurses-5.3**
- **Patch-2.5.4**
- **Sed-3.0.2**
- **Tar-1.22**
- **Texinfo-4.7**
- **XZ Utils-4.999.8beta**

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

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

# Simple script to list version numbers of critical development tools

bash --version | head -n1 | cut -d" " -f2-4
echo -n "Binutils: "; ld --version | head -n1 | cut -d" " -f3-
bison --version | head -n1
bzip2 --version 2>&1 < /dev/null | head -n1 | cut -d" " -f1,6-
echo -n "Coreutils: "; chown --version | head -n1 | cut -d")" -f2
diff --version | head -n1
find --version | head -n1
gawk --version | head -n1
gcc --version | head -n1
g++ --version | head -n1
ldd $(which ${SHELL}) | grep libc.so | cut -d ' ' -f 3 | ${SHELL} | head -n 1 | c
grep --version | head -n1
gzip --version | head -n1
make --version | head -n1
tic -V
patch --version | head -n1
sed --version | head -n1
tar --version | head -n1
makeinfo --version | head -n1
xz --version | head -n1
echo 'main(){}' | gcc -v -o /dev/null -x c - > dummy.log 2>&1
if ! grep -q ' error' dummy.log; then
  echo "Compilation successful" && rm dummy.log
else
  echo 1>&2  "Compilation FAILED - more development packages may need to be \
installed. If you like, you can also view dummy.log for more details."
fi
EOF

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

Typography

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

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

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

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

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

Emphasis

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

http://cross-lfs.org/

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

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

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

`[REPLACED TEXT]`

This format is used to encapsulate text that is not to be typed as seen or copied-and-pasted.

`passwd(5)`

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

Structure

This book is divided into the following parts.

Part I - Introduction

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

Part II - Preparing for the Build

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

Part III - Make the Cross-Compile Tools

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

Part IV - Building the Basic Tools

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

Part V - Building the CLFS System

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

Appendices

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

Errata

The software used to create a CLFS system is constantly being updated and enhanced. Security warnings and bug fixes may become available after the CLFS book has been released. Some host systems may also have problems building CLFS. To check whether the package versions or instructions in this release of CLFS need any modifications to accommodate security vulnerabilities, other bug fixes, or host-specific issues, please visit *http://trac.cross-lfs.org/wiki/errata* before proceeding with your build. You should note any changes shown and apply them to the relevant section of the book as you progress with building the CLFS system.

Part I. Introduction

Chapter 1. Introduction

1.1. Cross-LFS Acknowledgements

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

Our Leaders:

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

Our CLFS Team:

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

Outside the Development Team

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

The CLFS team would also like to acknowledge contributions of people from *clfs-dev@lists.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.
- Theo Schneider - Testing of the Linux Headers Package
- Martin Ward - Recommendations for Systemd and the Boot method, among other contributions
- William Zhou - Text updates and Typos

Former Team Members

- Joe Ciccone - Lead Developer.
- Nathan Coulson - Bootscripts.
- Jim Gifford - Lead Developer.
- Jeremy Huntwork - PowerPC, x86, Sparc builds.
- Karen McGuiness - Proofreader.
- Ryan Oliver - Build Process Developer.
- Alexander E. Patrakov - Udev/Hotplug Integration
- Jeremy Utley - Release Manager 1.x Series.

- Zack Winkles - Unstable book work.

The Linux From Scratch Project

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

Thank you all for your support.

1.2. How to Build a CLFS System

The CLFS system will be built by using a previously installed Unix system or Linux distribution (such as Debian, Fedora, openSUSE, or Ubuntu). This existing system (the host) will be used as a starting point to provide necessary programs, including a compiler, linker, and shell, to build the new system. Select the “development” option during the distribution installation to be able to access these tools.

As an alternative to installing an entire separate distribution onto your machine, you may wish to use a livecd. Most distributions provide a livecd, which provides an environment to which you can add the required tools onto, allowing you to successfully follow the instructions in this book. Remember that if you reboot the livecd you will need to reconfigure the host environment before continuing with your build.

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

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

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

After we build our “Cross-Tools”, we start building a very minimal working system in `/tools`, using the cross-toolchain in `/cross-tools`. Once the temporary system is finished, we perform a few additional tasks to prepare to enter this temporary build environment, either by booting or chrooting into it. For more details about the difference between these methods, see Section 6.31, “To Boot or to Chroot?”.

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

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

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

1.3. Master Changelog

This is version 3.0.0-SYSVINIT of the Cross-Compiled Linux From Scratch book, dated October 18, 2014. 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:

- 18 October 2014
  - [William Harrington] - CLFS 3.0.0 SYSVINIT released.
- 26 September 2014
  - [William Harrington] - Update Bash 4.3 branch update to patch level 30.
  - [William Harrington] - Update Vim 7.4 branch update to patch level 473.
- 09 October 2014
  - [William Harrington] - Update Linux Kernel to 3.14.21.
- 08 October 2014
  - [William Harrington] - Update Shadow to 4.2.1.
- 03 October 2014
  - [William Harrington] - Add Util-linux Pass 2.
- 26 September 2014
  - [William Harrington] - Update Bash 4.3 branch update to patch level 26.
  - [William Harrington] - Update Readline 6.3 branch update to patch level 8.
  - [William Harrington] - Update Linux sublevel patch to 3.14.19.
  - [William Harrington] - Update Vim 7.4 branch update patch to level 460.
- 18 August 2014
  - [Chris] - Removed obsolete --with-tls and --with-__thread options from Cross-Tools Glibc builds - those options have long since been removed from Glibc.
- 17 August 2014
  - [Chris] - Combined Makefile modification commands in Boot section Shadow install into one command.
  - [Chris] - Removed redundant command explanation from Less installation page.
- 14 August 2014
  - [Chris] - Removed obsolete command to disable test lib in Glibc instructions.
- 10 August 2014
  - [Chris] - Removed several switches from GCC builds for options that are enabled by default.
  - [Chris] - Removed --enable-multibyte switch from Vim pages, since multibyte support is enabled by default.
  - [Chris] - Removed --enable-shared switch from CLooG pages; shared libraries are built by default.
- 09 August 2014
  - [Chris] - Redo top index to list by arch, then 32/64/multilib within each arch.

  - [Chris] - Removed "M4=m4" from final-system Bison and Flex pages, since m4 is now installed before them.
- 07 August 2014
  - [Chris] - Updated Linux sublevel patch to 3.14.16.
- 05 August 2014
  - [Chris] - Reversed change to Vim's docs location, so that Vim can find its help files.
- 01 August 2014
  - [Chris] - Updated Linux sublevel to 3.14.15.
  - [Chris] - Build Pkg-config-lite before Ncurses in the final system, so that Ncurses will install .pc files.
  - [Chris] - Move Bzip2 before Perl and modify Perl instructions to use system-install Bzip2. Solution borrowed from LFS.
- 28 July 2014
  - [Chris] - Removed unneeded notes about the importance of the testuites for GMP, MPFR, MPC, ISL, and CLooG.
- 24 July 2014
  - [Chris] - Added config flag descriptions to kernel config pages. Thanks to Hazel Russman for the idea.
  - [Chris] - Removed commands to recreate /usr/share/info to reduce possible user confusion.
- 21 July 2014
  - [William Harrington] - Update MPFR 3.1.2 fixes patch for patch level 10 update.
  - [William Harrington] - Update VIM 7.4 Branch Update patch to patch level 373.
  - [William Harrington] - Update Linux to 3.14.13.
- 18 July 2014
  - [Chris] - Changed title on Linux Headers installation pages to reduce potential confusion with previous Linux-Headers package.
- 14 July 2014
  - [Chris] - Modified instructions for mounting file systems for chroot so that they are all mounted before entering chroot, and added a tmpfs on /run. Took solution for /dev/shm symlink from LFS. Fixes ticket *#971*.
  - [Chris] - Updated Linux sublevel to 3.14.12.
- 10 July 2014
  - [Chris] - Removed creation of /var/run/utmp - it's created on boot in the bootscripts.
  - [Chris] - Removed creation of pidof from Sysvinit installation, to use pidof from Procps-ng instead.
- 02 July 2014
  - [William Harrington] - Update Linux subelevel patch to 3.14.10.
- 30 June 2014
  - [Chris] - Added documentation installation instructions for Zlib.
- 27 June 2014
  - [William Harrington] - Updated KMOD to 18. Fixes ticket *#965*.

- 20 June 2014
  - [William Harrington] - Remove **make -C man install** from final-system KMOD as man pages are installed during **make install**.
  - [William Harrington] - Add Linux sublevel patch.
  - [Chris] - Changed GMP documentation installation instructions to use the Makefile to install docs instead of doing so manually.
- 17 June 2014
  - [William Harrington] - Use --with-isl=system for temp-system CLooG.
- 17 June 2014
  - [William Harrington] - Move Man-DB right after Libpipeline.
- 15 June 2014
  - [Chris] - Added commands to create and install HTML documentation for MPC.
- 13 June 2014
  - [Chris] - Text updates to kernel installation page.
  - [William Harrington - Update File to 5.19.
- 11 June 2014
  - [Chris] - Placed file before findutils in temp-system.
- 10 June 2014
  - [William Harrington] - Upgraded Gettext to 0.19.1.
- 04 June 2014
  - [Chris] - Added Introduction page to final-preps.
- 03 June 2014
  - [Chris] - Moved installation of /tools/include/libiberty.h from Cross-Tools to Temp-System.
  - [Chris] - Moved creation of /var/run symlink to a more appropriate location in the Creating Directories page.
- 02 June 2014
  - [William Harrington] - Upgraded Gettext to 0.19.
- 01 June 2014
  - [Chris] - Upgraded GCC to 4.8.3.
  - [William Harrington] - Upgraded Eudev to 1.7.
- 31 May 2014
  - [William Harrington] - Update Check to 0.9.13.
  - [William Harrington] - Update Linux to 3.14.5.
  - [William Harrington] - Update Vim 7.4 branch update patch to level 316.
- 30 May 2014
  - [William Harrington] - Add --host=${CLFS_TARGET} during cross-tools Pkg-config configure.
- 29 May 2014

  - [William Harrington] - Update Perl to 5.20.0.
  - [Chris] - Edited "how" page - removed details about booting/chrooting and moved them to the "choose" page for temp-system.
- 28 May 2014
  - [William Harrington] - Upgrade TZData to 2014d.
  - [William Harrington] - Upgrade Man-pages to 3.68.
- 25 May 2014
  - [William Harrington] - Upgrade Grep to 2.19.
- 24 May 2014
  - [Chris] - Moved Bc from Cross-Tools to Boot section, as it's only needed to compile the kernel.
  - [Chris] - Moved "About ${CLFS}" page from final-preps to partitioning - best to have the page explaining ${CLFS} before it's actually used.
  - [Chris] - Added creation of log files with passwd and group.
- 22 May 2014
  - [Chris] - Removed xgettext and msgmerge from temp-system Gettext - those are only needed for ACL/Attr.
  - [Chris] - Swapped build order of Linux-headers and File in Cross-Tools - no particular reason for File to be after Linux-headers, so just make them alphabetical.
- 21 May 2014
  - [Chris] - Renamed Bootscripts chapter back to System Configuration - half that section doesn't involve bootscripts, and this allows for more consistency between different books .
  - [Chris] - Rewrote Eudev page in system-config section.
  - [William Harrington] - Update Man-pages to 3.67.
  - [Chris] - Moved creation of /etc/fstab to system-config.
- 19 May 2014
  - [Chris] - Added commands to unset CFLAGS and CXXFLAGS to the clfs user's default .bashrc, and removed the separate page where this was done.
  - [Chris] - Removed separate "Build flags" pages and moved that info to "Build Variables" pages, and moved those from cross-tools to final-preps.
- 18 May 2014
  - [Chris] - Added Pkg-config-lite to Cross-Tools and removed PKG_CONFIG parameters from temp-system and boot packages.
- 17 May 2014
  - [William Harrington] - Upgrade TZData to 2014c.
  - [William Harrington] - Add --disable-werror to cross-tools Bintuils instructions.
  - [William Harrington] - Move **logger** to `/bin` for CLFS-Bootscripts requirement.
  - [Chris] - Changed "systemd" reference on Kbd page to CLFS Bootscripts, and added dumpkeys and kbd_mode to list of programs moved to /bin as they're used in the bootscripts.
- 14 May 2014

  - [Chris] - Created Sysvinit branch - removed Attr, ACL, D-Bus, Expat, Gperf, Intltool, Libcap, Systemd, XML::Parser, and added Libee, Libestr, Rsyslog.
- 11 May 2014
  - [Chris] - Added explanations for Essential Symlinks.
- 08 May 2014
  - [William Harrington] - Update Man-pages to 3.66.
  - [William Harrington] - Add GDBM 1.11 to the books.
  - [William Harrington] - Add Libpipeline 1.3.0 to the books.
  - [William Harrington] - Add Man-DB 2.6.7.1 to the books.
  - [William Harrington] - Remove Man-1.6g from the books.
  - [William Harrington] - Remove zsoelim link from Groff.
- 06 May 2014
  - [Chris] - Removed unneeded configure switches specifying GMP/MPFR/MPC/ISL/CLOOG in /tools.
  - [Chris] - Removed unneeded --libexecdir switches from temp-system GCC.
- 02 May 2014
  - [Chris] - Removed command to copy gconv-modules in Glibc instructions, as it is no longer needed.
- 30 April 2014
  - [Chris] - Added "PKG_CONFIG=" to boot section kmod commands, otherwise it will use pkg-config to find the "bash-completion" package and install stuff to that prefix if found. Thanks to boloco in IRC for finding this.
  - [William Harrington] - Uprade DBus to 1.8.2.
- 28 April 2014
  - [Chris] - Remove all mentions of Alpha architecture, as it has not been tested for a long time.
- 27 April 2014
  - [William Harrington] - Update IPRoute2 to 3.14.0.
- 27 April 2014
  - [Chris] - Updated Acknowledgements page.
  - [William Harrington] - Update Linux to 3.14.2.
- 26 April 2014
  - [Chris] - Renamed bootscripts package to boot-scripts (thanks to William Harrington for the suggestion) and removed the patch as it is not needed for the new package.
  - [Chris] - Renamed bootscripts chapter to "System Configuration".
  - [William Harrington] - Update Util-linux to 2.24.2.
  - [Chris] - Added new page about how to view the book from inside the temporary boot environment.
  - [Chris] - Don't move binaries from Findutils or Psmisc to /bin, as they're no longer needed there.
  - [William Harrington] - Update Vim 7.4 branch update patch to level 265.
- 24 April 2014

  - [William Harrington] - Update Eudev to 1.6.
  - [William Harrington] - Update Man-pages to 3.65.
  - [William Harrington] - Update Bash branch update patch to level 11.
  - [William Harrington] - Update Readline branch update patch to level 5.
- 22 April 2014
  - [Chris] - Removed several virtual filesystems from ${CLFS}/fstab for the boot method, as they're already mounted by the bootscript.
  - [Chris] - Removed creation of gtbl and geqn symlinks, as they are no longer needed.
- 21 April 2014
  - [Chris] - Re-added manual creation of /usr/bin/perl symlink, as it is not created automatically under some circumstances.
- 20 April 2014
  - [Chris] - Reorganized "boot" section to allow everything to be installed into /tools, remove the need to chown ${CLFS} to the clfs user, and generally make the chapter's instructions flow better.
- 18 April 2014
  - [Chris] - Added --disable-makeinstall-setuid to temp-system Util-linux, as nothing there needs to be suid.
- 16 April 2014
  - [Chris] - Removed creation of /usr/bin/perl symlink, as Perl creates a link itself when installed.
- 13 April 2014
  - [Chris] - Updated most packages in the Boot section to install into /tools instead of $CLFS. Thanks to Martin Ward and William Harrington for assistance.
- 12 April 2014
  - [Chris] - Modified sysvinit installation to allow it to use /tools/etc/inittab and /tools/sbin/agetty.
  - [Chris] - Moved Util-linux installation out of boot/chroot sections into the temp-system, as it can be installed the same way with either method.
- 11 April 2014
  - [Chris] - Updated MPC to 1.0.2.
  - [Chris] - Removed unneeded --disable-login and --disable-su switches from Util-linux configure in boot/chroot sections.
- 09 April 2014
  - [William Harrington] - Update Eudev to 1.5.3.
- 09 April 2014
  - [Chris] - Remove obsolete --enable-cloog-backend parameter from GCC configure.
  - [William Harrington] - Remove unneeded --disable-isl-version-check parameter from GCC configure.
  - [William Harrington] - Update Gawk to 4.1.1.
- 08 April 2014
  - [William Harrington] - Add versioned doc directories for Automake, DBus, Flex, Gawk, Gettext, GMP, IPRoute2, KBD, MPFR, Readline, Sed, Tar, Util-linux and XZ.

- 06 April 2014
  - [William Harrington] - Update KMOD to 17.
  - [William Harrington] - Update Man-pages to 3.64.
- 05 April 2014
  - [William Harrington] - Move Flex before Bison in final system and remove bison and flex from temp system.
  - [William Harrington] - Add M4=m4 before configure command in final-system Bison and Flex.
  - [William Harrington] - Remove Bison and Flex from temp system.
  - [William Harrington] - Move M4 before GMP in final-system and remove M4 from temp system.
- 02 April 2014
  - [William Harrington] - Add test commands to Attr and Acl.
- 30 March 2014
  - [Chris] - Modified network section to split systemd/sysconfig network configurations.
  - [Chris] - Removed instructions to install systemd manpages - they are now automatically installed by default.
  - [Chris] - Updated installed program lists for several packages.
  - [William Harrington] - Remove link of libl to libfl during Flex installation.
  - [William Harrington] - Update DHCPCD to 6.3.2.
  - [William Harrington] - Update VIM 7.4 patch to level 229.
  - [William Harrington] - Update Bash 4.3 patch to level 8.
  - [William Harrington] - Update Readline 6.3 patch to level 3.
  - [William Harrington] - Update GCC 4.8.2 branch update patch to revision 208943.
- 23 March 2014
  - [William Harrington] - Update Man-pages to 3.63.
- 28 March 2014
  - [William Harrington] - Update File to 5.18.
  - [William Harrington] - Update Flex to 2.5.39.
  - [William Harrington] - Update Linux to 3.12.15.
  - [William Harrington] - Update Systemd to 212.
  - [William Harrington] - Update Tzdata to 2014b.
  - [William Harrington] - Update GMP to 6.0.0a.
  - [William Harrington] - Move Bison and Flex before Binutils.
- 17 March 2014
  - [William Harrington] - Remove libdbus and Update Systemd to 211.
  - [William Harrington] - Add Systemd compat patch to install pkg-config files.
- 12 March 2014
  - [Chris] - Updated documentation installation instructions for Readline and Bash.
  - [Chris] - Deleted --remove-destination parameter when creating /etc/localtime - it's no longer needed.

- [William Harrington] - Update Man-pages to 3.62.
- 10 March 2014
  - [William Harrington] - Update TZData to 2014a.
  - [William Harrington] - Update MPFR fixes patch to patch level 5.
- 09 March 2014
  - [William Harrington] - Add configure command descriptions and testsuite commands to Systemd.
  - [William Harrington] - Update testsuite description for D-BUS.
  - [William Harrington] - Add note to final-system Util-Linux regarding extra functionality with libudev.
  - [William Harrington] - Add commands and descriptions for `/etc/adjtime`.
- 02 March 2014
  - [William Harrington] - Update DHCPCD to 6.3.1.
- 27 February 2014
  - [William Harrington] - Update Grep to 2.18.
  - [William Harrington] - Update DHCPCD to 6.3.0.
  - [William Harrington] - Update Man-pages to 3.61.
  - [William Harrington] - Update Readline to 6.3.
  - [William Harrington] - Update Bash to 4.3.
- 24 February 2014
  - [William Harrington] - Remove installation of nscd tmpfile and unit service file in clfs-network-scripts.
- 23 February 2014
  - [William Harrington] - Add commands for **nscd** config, runtime, and Systemd unit files installation to GLIBC.
- 18 February 2014
  - [William Harrington] - Migrate EGLIBC to GLIBC and upgrade to version 2.19.
  - [William Harrington] - Update D-Bus to stable version 1.8.0.
  - [William Harrington] - Update File to version 5.17.
  - [William Harrington] - Update Grep to version 2.17.
  - [William Harrington] - Update Linux to version 3.12.11.
  - [William Harrington] - Update Man-pages to version 3.60.
  - [William Harrington] - Update Psmisc to version 22.21.
- 13 February 2014
  - [William Harrington] - Correct nobody gid to use nogroup gid.
- 03 February 2014
  - [William Harrington] - Add configure options to systemd so loadkeys and setfont is searched in /bin instead of /usr/bin.
- 02 February 2014

  - [William Harrington] - Remove note in Iana-etc page and update iana-etc update numbers patch for proper operation with current iana-etc format.
- 29 January 2014
  - [William Harrington] - Add PKG_CONFIG= to temp-system make configure command.
- 27 January 2014
  - [William Harrington] - Add PERL=/usr/bin/perl to temp-system Texinfo build.
- 26 January 2014
  - [William Harrington] - Update LESS to 462.
- 25 January 2014
  - [William Harrington] - Create a link from /proc/self/mounts to /etc/mtab in boot and chroot createfiles section.
- 23 January 2014
  - [William Harrington] - Update text and redo some commands regarding Systemd in the System and Network configuration chapters.
  - [William Harrington] - Add a clock configuration script to system configuration chapter with regards to systemd.
  - [William Harrington] - Add CLFS-Network-Scripts and add commands to install them in the network configuration chapter.
  - [William Harrington] - Update Util-linux to 2.24.1.
  - [William Harrington] - Update Check to 0.9.12.
- 21 January 2014
  - [William Harrington] - Add D-Bus libraries before Systemd in final-system for a circular dependency issue.
- 19 January 2014
  - [William Harrington] - Update DHCPCD to 6.2.1.
  - [William Harrington] - Use readlink when creating links for dynamic libraries for zlib, xz, ncurses, and readline.
- 13 January 2014
  - [William Harrington] - Update ISL to 0.12.2.
  - [William Harrington] - Update Man-pages to 3.56.
- 10 January 2014
  - [William Harrington] - Migrate Procps to Procps-ng.
- 09 January 2014
  - [Chris] - Updated Linux to 3.12.7.
  - [Chris] - Updated Perl to 5.18.2.
  - [Chris] - Updated Gettext to 0.18.3.2.
  - [Chris] - Updated Libcap to 2.24.
  - [Chris] - Removed unneeded command to create a /usr/lib/terminfo symlink from Ncurses instructions.
  - [Chris] - Removed /usr/{doc,info,man} symlinks and edited instructions for packages to ensure all documentation is installed to /usr/share.

  - [Chris] - Moved libcap so it's built just before iputils, which can use libcap.
- 08 January 2014
  - [Chris] - Removed commands to create files in /var/run and /var/log - systemd now used its journal for all logs.
  - [Chris] - Removed obsolete sed command from temp-system Coreutils.
  - [Chris] - Removed unneeded command to rename stubs-64.h.
  - [Chris] - Removed unneeded "make configure-host" commands from Binutils.
  - [Chris] - Moved Attr and ACL to just after GCC in the final system.
- 07 January 2014
  - [Chris] - Added command, from LFS, to modify /tools/lib/libstdc++.la to prevent a /tools reference in the final system.
  - [Chris] - Use libiberty from GCC, since Binutils no longer installs it by default.
- 06 January 2014
  - [Chris] - Added msgmerge and xgettext to /tools Gettext installation - needed for attr.
  - [William Harrington] - Update DHCPCD to 6.2.0.
  - [William Harrington] - Update Grep to 2.16.
- 05 January 2014
  - [Chris] - Removed rsyslog and its depedencies, libee and libestr.
  - [Chris] - Updated Expat to 2.1.0.
  - [Chris] - Updated Attr to 2.4.47.
  - [Chris] - Updated D-Bus to 1.6.18.
  - [Chris] - Updated Libcap to 2.23.
- 04 January 2014
  - [Chris] - Updated Systemd to 208.
  - [Chris] - Added Gperf, XML::Parser, Intltool, and ACL - needed by systemd.
- 01 January 2014
  - [Chris] - Added command to EGLIBC instructions to remove extraneous '.x' files from /usr/include/rpcsvc.
- 23 December 2013
  - [William Harrington] - Update E2fsprogs to 1.42.9.
  - [William Harrington] - Update Automake to 1.14.1.
- 21 December 2013
  - [William Harrington] - Update CLooG to 0.18.2.
  - [William Harrington] - Update Tzdata to 2013i.
  - [William Harrington] - Update EGLIBC 2.18 to revision 24829.
  - [William Harrington] - Update Linux to 3.12.6.
- 14 December 2013

  - [William Harrington] - Update Coreutils to 8.22.
  - [William Harrington] - Update Man-pages to 3.55.
- 08 December 2013
  - [William Harrington] - Update Bison to 3.0.2.
- 06 December 2013
  - [Chris] - Modified XZ instructions so that it won't install liblzma.{a,la} in /lib.
- 04 December 2013
  - [William Harrington] - Updated Binutils to version 2.24.
  - [William Harrington] - Updated File to version 5.16.
- 26 November 2013
  - [Chris] - Updated installed program lists for several packages.
  - [Chris] - Removed redundant --disable-su and --disable-login switches from Util-linux - they aren't built anyway if PAM isn't found.
- 25 November 2013
  - [William Harrington] - Update Iproute2 to 3.12.0.
  - [William Harrington] - Update Tar to 1.27.1.
- 23 November 2013
  - [Chris] - Use nologin from Util-linux instead of Shadow.
- 22 November 2013
  - [Chris] - Util-linux now installs last and mesg by default, so use those instead of the versions in Sysvinit.
- 18 November 2013
  - [William Harrington] - Move creation of dummy groups and user to final-system Util-linux.
  - [William Harrington] - Remove SUBDIRS variable from final-system Coreutils test command.
  - [William Harrington] - Adjust Util-linux test commands.
- 13 November 2013
  - [William Harrington] - Adjust Util-linux check commands.
  - [William Harrington] - Move Shadow before Util-Linux in final-system.
  - [William Harrington] - Update Bison to 3.0.1.
  - [William Harrington] - Remove Bison 3.0 eof patch.
- 11 November 2013
  - [William Harrington] - Update TZDATA to 2013h.
  - [William Harrington] - Add MPFR 3.1.2 Fixes patch.
  - [William Harrington] - Update Grep to 2.15.
  - [William Harrington] - Update Linux to 3.12.
  - [William Harrington] - Add a sed for Texinfo 5.x to all sections of Binutils.
  - [William Harrington] - Update Texinfo to 5.2.

  - [William Harrington] - Update KBD to 2.0.1.
  - [William Harrington] - Update Check to 0.9.11.
  - [William Harrington] - Update Automake to 1.14.
  - [William Harrington] - Update Make to 4.0.
  - [William Harrington] - Add a Sed command to EGLIBC for Make-4.x.
  - [William Harrington] - Move Check to temp-system.
- 09 November 2013
  - [William Harrington] - Add PKG_CONFIG= in front of configure line for chroot and boot Util-linux for hosts with pkg-config installed.
- 25 October 2013
  - [William Harrington] - Updated Util-Linux to 2.24.
  - [William Harrington] - Updated Tar to 1.27.
  - [William Harrington] - Updated Linux to 3.10.17.
  - [William Harrington] - Updated GCC to 4.8.2.
  - [William Harrington] - Updated CLooG to 0.18.1.
  - [William Harrington] - Changelog restarted, see the 2.1.0 book for the old changelog.

1.4. Changelog for PowerPC

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

Changelog Entries:

- 03 February 2014
  - [William Harrington] - Add --mandir=/usr/share/man to HFSUtils configure command.
- 24 October 2013
  - [William Harrington] - Changelog restarted, see the 2.1.0 book for the old changelog.

1.5. Resources

1.5.1. FAQ

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

1.5.2. Mailing Lists

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

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

For information on the different lists, how to subscribe, archive locations, and additional information, visit *http://trac.cross-lfs.org/wiki/lists*.

1.5.3. News Server

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

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

1.5.4. IRC

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

1.5.5. Mirror Sites

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

1.5.6. Contact Information

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

1.6. Help

If an issue or a question is encountered while working through this book, check the FAQ page at *http://trac.cross-lfs.org/wiki/faq#generalfaq*. Questions are often already answered there. If your question is not answered on this page, try to find the source of the problem. The following hint will give you some guidance for troubleshooting: *http://hints.cross-lfs.org/index.php/Errors*.

We also have a wonderful CLFS community that is willing to offer assistance through the mailing lists and IRC (see the Section 1.5, “Resources” section of this book). However, we get several support questions everyday and many of them can be easily answered by going to the FAQ and by searching the mailing lists first. So for us to offer the best assistance possible, you need to do some research on your own first. This allows us to focus on the more unusual support needs. If your searches do not produce a solution, please include all relevant information (mentioned below) in your request for help.

1.6.1. Things to Mention

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

- The version of the book being used (in this case 3.0.0-SYSVINIT)

- The host distribution and version being used to create CLFS.
- The architecture of the host and target.
- The value of the ${CLFS_HOST} and ${CLFS_TARGET} environment variables, and if applicable, ${BUILD32}, ${BUILD64}, ${BUILDN32}, and ${GCCTARGET}.
- The package or section in which the problem was encountered.
- The exact error message or symptom received. See Section 1.6.3, "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. Setting The ${CLFS} Variable

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

```
export CLFS=[/mnt/clfs]
```

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

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

```
echo ${CLFS}
```

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

2.3. Creating a New Partition

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

A minimal system requires around 6 gigabytes (GB). This is enough to store all the source tarballs and compile the packages. The CLFS system itself will not take up this much room. A large portion of this requirement is to provide sufficient free temporary storage. Compiling packages can require a lot of disk space which will be reclaimed after the package is installed. If the CLFS system is intended to be the primary Linux system, additional software will probably be installed which will require additional space (2-10 GB).

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

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

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

Remember the designation of the new partition (e.g., `hda5`). This book will refer to this as the CLFS partition. Also remember the designation of the swap partition. These names will be needed later for the `/etc/fstab` file. You will also need to know the designation of the apple_bootstrap partition for the `yaboot.conf` when you set this up before you run **ybin**.

2.4. Creating a File System on the Partition

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

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

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

Replace *`[xxx]`* with the name of the CLFS partition (`sda5` in our previous example).

Note

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

```
debugfs -R feature /dev/[xxx]
```

If the output contains features other than: dir_index; filetype; large_file; resize_inode or sparse_super then your host system may have custom enhancements. In that case, to avoid later problems, you should compile the stock E2fsprogs package and use the resulting binaries to re-create the filesystem on your CLFS partition. To do this, run the following commands as `root`:

```
cd /tmp
tar xjf /path/to/sources/e2fsprogs-1.42.9.tar.bz2
cd e2fsprogs-1.42.9
mkdir build
cd build
../configure
make #note that we intentionally don't 'make install' here!
./misc/mke2fs /dev/[xxx]
cd /tmp
rm -rf e2fsprogs-1.42.9
```

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

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

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

2.5. Mounting the New Partition

Now that a file system has been created, the partition needs to be made accessible. In order to do this, the partition needs to be mounted at a chosen mount point.

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

```
export CLFS=[/mnt/clfs]
```

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

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

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

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

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

Replace *[xxx]* and *[yyy]* with the appropriate partition names.

Ensure that this new partition is not mounted with permissions that are too restrictive (such as the `nosuid`, `nodev`, or `noatime` options). Run **mount | grep ${CLFS}** to see what options are set for the mounted CLFS partition. If `nosuid`, `nodev`, and/or `noatime` are set, the partition will need to be remounted.

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

Chapter 3. Packages and Patches

3.1. Introduction

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

Download locations may not always be accessible. If a download location has changed since this book was published, Google (*http://www.google.com/*) provides a useful search engine for most packages. If this search is unsuccessful, try one of the alternative means of downloading discussed at *http://cross-lfs.org/files/packages/3.0.0/SYSVINIT/*.

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

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

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

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

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

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

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

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

MD5SUMS:

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

SHA1SUMS:

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

3.2. All Packages

Download or otherwise obtain the following packages:

- **Autoconf (2.69) - 1,188 KB:**
Home page: *http://www.gnu.org/software/autoconf*
Download: *http://ftp.gnu.org/gnu/autoconf/autoconf-2.69.tar.xz*
MD5 sum: `50f97f4159805e374639a73e2636f22e`

• **Automake (1.14.1) - 1,489 KB:**
Home page: *http://www.gnu.org/software/automake*
Download: *http://ftp.gnu.org/gnu/automake/automake-1.14.1.tar.xz*
MD5 sum: `7fc29854c520f56b07aa232a0f880292`

• **Bash (4.3) - 7,956 KB:**
Home page: *http://www.gnu.org/software/bash*
Download: *http://ftp.gnu.org/gnu/bash/bash-4.3.tar.gz*
MD5 sum: `81348932d5da294953e15d4814c74dd1`

• **Bc (1.06.95) - 284 KB:**
Home page: *http://www.gnu.org/software/bc/*
Download: *ftp://alpha.gnu.org/gnu/bc/bc-1.06.95.tar.bz2*
MD5 sum: `5126a721b73f97d715bb72c13c889035`

• **Binutils (2.24) - 22,717 KB:**
Home page: *http://sources.redhat.com/binutils*
Download: *http://ftp.gnu.org/gnu/binutils/binutils-2.24.tar.bz2*
MD5 sum: `e0f71a7b2ddab0f8612336ac81d9636b`

• **Bison (3.0.2) - 1,927 KB:**
Home page: *http://www.gnu.org/software/bison*
Download: *http://ftp.gnu.org/gnu/bison/bison-3.0.2.tar.xz*
MD5 sum: `146be9ff9fbd27497f0bf2286a5a2082`

• **Bootscripts for CLFS (3.0-20140710) - 41 KB:**
Download: *http://ftp.cross-lfs.org/pub/clfs/conglomeration/bootscripts-cross-lfs/bootscripts-cross-lfs-3.0-20140710.tar.xz*
MD5 sum: `ebe69a3adc1da12bfcdfab3e094eeb1a`

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

• **Check (0.9.13) - 753 KB:**
Home page: *http://check.sourceforge.net/*
Download: *http://sourceforge.net/projects/check/files/check/0.9.13/check-0.9.13.tar.gz*
MD5 sum: `95530868f81a9496b2518fd2b713008a`

• **ClooG (0.18.2) - 2,377 KB:**
Home page: *http://cloog.org*
Download: *http://www.bastoul.net/cloog/pages/download/cloog-0.18.2.tar.gz*
MD5 sum: `69116aa6cd5e73f6b688d871875e1292`

• **Coreutils (8.22) - 5,335 KB:**
Home page: *http://www.gnu.org/software/coreutils*
Download: *http://ftp.gnu.org/gnu/coreutils/coreutils-8.22.tar.xz*
MD5 sum: `8fb0ae2267aa6e728958adc38f8163a2`

• **DejaGNU (1.5.1) - 568 KB:**
Home page: *http://www.gnu.org/software/dejagnu*
Download: *http://ftp.gnu.org/gnu/dejagnu/dejagnu-1.5.1.tar.gz*
MD5 sum: `8386e04e362345f50ad169f052f4c4ab`

- **DHCPCD (6.3.2) - 137 KB:**

Home page: *http://roy.marples.name/projects/dhcpcd*
Download: *http://roy.marples.name/downloads/dhcpcd/dhcpcd-6.3.2.tar.bz2*
MD5 sum: `bb7d8e8533161fbe0eee3b8830fdbdf4`

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

Home page: *http://www.gnu.org/software/diffutils*
Download: *http://ftp.gnu.org/gnu/diffutils/diffutils-3.3.tar.xz*
MD5 sum: `99180208ec2a82ce71f55b0d7389f1b3`

- **E2fsprogs (1.42.9) - 4,656 KB:**

Home page: *http://e2fsprogs.sourceforge.net*
Download: *http://www.kernel.org/pub/linux/kernel/people/tytso/e2fsprogs/v1.42.9/e2fsprogs-1.42.9.tar.xz*
MD5 sum: `55cc59c587a7199fd9d93b2843842236`

- **Eudev (1.7) - 1,756 KB:**

Home page: *http://www.gentoo.org/proj/en/eudev/*
Download: *http://dev.gentoo.org/~blueness/eudev/eudev-1.7.tar.gz*
MD5 sum: `80649a0350ff9620fc2da9562d9f2a6a`

- **Expect (5.45) - 616 KB:**

Home page: *http://expect.sourceforge.net*
Download: *http://downloads.sourceforge.net/project/expect/Expect/5.45/expect5.45.tar.gz*
MD5 sum: `44e1a4f4c877e9ddc5a542dfa7ecc92b`

- **File (5.19) - 719 KB:**

Home page: *http://www.darwinsys.com/file*
Download: *ftp://ftp.astron.com/pub/file/file-5.19.tar.gz*
MD5 sum: `e3526f59023f3f7d1ffa4d541335edab`

Note

File (5.19) 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/3.0.0/SYSVINIT/*.

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

Home page: *http://www.gnu.org/software/findutils*
Download: *http://ftp.gnu.org/gnu/findutils/findutils-4.4.2.tar.gz*
MD5 sum: `351cc4adb07d54877fa15f75fb77d39f`

- **Flex (2.5.39) - 1,612 KB:**

Home page: *http://flex.sourceforge.net*
Download: *http://downloads.sourceforge.net/flex/flex-2.5.39.tar.bz2*
MD5 sum: `77d44c6bb8c0705e0017ab9a84a1502b`

- **Gawk (4.1.1) - 2,197 KB:**

Home page: *http://www.gnu.org/software/gawk*
Download: *http://ftp.gnu.org/gnu/gawk/gawk-4.1.1.tar.xz*
MD5 sum: `a2a26543ce410eb74bc4a508349ed09a`

- **GCC (4.8.3) - 84,108 KB:**
Home page: *http://gcc.gnu.org*
Download: *ftp://gcc.gnu.org/pub/gcc/releases/gcc-4.8.3/gcc-4.8.3.tar.bz2*
MD5 sum: `7c60f24fab389f77af203d2516ee110f`

- **GDBM (1.11) - 812 KB:**
Home page: *http://www.gnu.org/software/gdbm*
Download: *http://ftp.gnu.org/gnu/gdbm/gdbm-1.11.tar.gz*
MD5 sum: `72c832680cf0999caedbe5b265c8c1bd`

- **Gettext (0.19.1) - 16,886 KB:**
Home page: *http://www.gnu.org/software/gettext*
Download: *http://ftp.gnu.org/gnu/gettext/gettext-0.19.1.tar.gz*
MD5 sum: `8949a57e82abe88274a93174ade515e4`

- **Glibc (2.19) - 12,083 KB:**
Home page: *http://www.gnu.org/software/libc/*
Download: *http://ftp.gnu.org/gnu/glibc/glibc-2.19.tar.xz*
MD5 sum: `e26b8cc666b162f999404b03970f14e4`

- **GMP (6.0.0) - 1,904 KB:**
Home page: *http://gmplib.org/*
Download: *http://ftp.gnu.org/gnu/gmp/gmp-6.0.0a.tar.xz*
MD5 sum: `1e6da4e434553d2811437aa42c7f7c76`

- **Grep (2.19) - 1,235 KB:**
Home page: *http://www.gnu.org/software/grep*
Download: *http://ftp.gnu.org/gnu/grep/grep-2.19.tar.xz*
MD5 sum: `ac732142227d9fe9567d71301e127979`

- **Groff (1.22.2) - 3,928 KB:**
Home page: *http://www.gnu.org/software/groff*
Download: *http://ftp.gnu.org/gnu/groff/groff-1.22.2.tar.gz*
MD5 sum: `9f4cd592a5efc7e36481d8d8d8af6d16`

- **Gzip (1.6) - 812 KB:**
Home page: *http://www.gnu.org/software/gzip/gzip.html*
Download: *http://ftp.gnu.org/gnu/gzip/gzip-1.6.tar.xz*
MD5 sum: `da981f86677d58a106496e68de6f8995`

- **Iana-Etc (2.30) - 204 KB:**
Home page: *http://www.archlinux.org/packages/core/any/iana-etc/*
Download: *http://ftp.cross-lfs.org/pub/clfs/conglomeration/iana-etc/iana-etc-2.30.tar.bz2*
MD5 sum: `3ba3afb1d1b261383d247f46cb135ee8`

- **IPRoute2 (3.14.0) - 436 KB:**
Home page: *http://www.linuxfoundation.org/collaborate/workgroups/networking/iproute2*
Download: *http://www.kernel.org/pub/linux/utils/net/iproute2/iproute2-3.14.0.tar.xz*
MD5 sum: `bd9d7567bbb987c88120669f5e1a1092`

- **IPutils (s20121221) - 155 KB:**
Home page: *http://www.linuxfoundation.org/en/Net:Iputils*
Download: *http://www.skbuff.net/iputils/iputils-s20121221.tar.bz2*
MD5 sum: `6072aef64205720dd1893b375e184171`
- **ISL (0.12.2) - 1,171 KB:**
Home page: *http://freecode.com/projects/isl*
Download: *http://isl.gforge.inria.fr/isl-0.12.2.tar.lzma*
MD5 sum: `b7d59eb79d3884e14b9788cc396f7687`
- **Kbd (2.0.1) - 951 KB:**
Home page: *http://kbd-project.org/*
Download: *http://kbd-project.org/download/kbd-2.0.1.tar.xz*
MD5 sum: `55453cb09d90370005e696949092b8f6`
- **Kmod (18) - 1,474 KB:**
Home page: *http://git.kernel.org/?p=utils/kernel/kmod/kmod.git;a=summary*
Download: *http://www.kernel.org/pub/linux/utils/kernel/kmod/kmod-18.tar.xz*
MD5 sum: `82835c7f01983634e06ca72b4ee30cc6`
- **Less (462) - 312 KB:**
Home page: *http://www.greenwoodsoftware.com/less*
Download: *http://www.greenwoodsoftware.com/less/less-462.tar.gz*
MD5 sum: `55a001838501ed8a129682de047960df`
- **Libee (0.4.1) - 352 KB:**
Home page: *http://www.libee.org/*
Download: *http://www.libee.org/download/files/download/libee-0.4.1.tar.gz*
MD5 sum: `7bbf4160876c12db6193c06e2badedb2`
- **Libestr (0.1.5) - 326 KB:**
Home page: *http://libestr.adiscon.com/*
Download: *http://libestr.adiscon.com/files/download/libestr-0.1.5.tar.gz*
MD5 sum: `f180c0cdc82883d161eba3f2e8a34eb4`
- **Libpipeline (1.3.0) - 787 KB:**
Home page: *http://libpipeline.nongnu.org/*
Download: *http://download.savannah.gnu.org/releases/libpipeline/libpipeline-1.3.0.tar.gz*
MD5 sum: `242428c01dca255cdcb2195073a9c6ed`
- **Libtool (2.4.2) - 852 KB:**
Home page: *http://www.gnu.org/software/libtool*
Download: *http://ftp.gnu.org/gnu/libtool/libtool-2.4.2.tar.xz*
MD5 sum: `2ec8997e0c07249eb4cbd072417d70fe`
- **Linux (3.14) - 78,399 KB:**
Home page: *http://www.kernel.org*
Download: *http://www.kernel.org/pub/linux/kernel/v3.0/linux-3.14.tar.xz*
MD5 sum: `b621207b3f6ecbb67db18b13258f8ea8`

- **M4 (1.4.17) - 1,149 KB:**
Home page: *http://www.gnu.org/software/m4*
Download: *http://ftp.gnu.org/gnu/m4/m4-1.4.17.tar.xz*
MD5 sum: `12a3c829301a4fd6586a57d3fcf196dc`

- **Make (4.0) - 1,342 KB:**
Home page: *http://www.gnu.org/software/make*
Download: *http://ftp.gnu.org/gnu/make/make-4.0.tar.bz2*
MD5 sum: `571d470a7647b455e3af3f92d79f1c18`

- **Man-DB (2.6.7.1) - 1,452 KB:**
Download: *http://cross-lfs.org/files/packages/3.0.0/SYSVINIT/man-db-2.6.7.1.tar.xz*
MD5 sum: `ce7b697f8e8016a085d9f5975ae6c4fb`

- **Man-pages (3.68) - 1,254 KB:**
Home page: *http://www.win.tue.nl/~aeb/linux/man*
Download: *http://www.kernel.org/pub/linux/docs/man-pages/man-pages-3.68.tar.xz*
MD5 sum: `43965ed65b34aeff75767b162f97eb01`

- **MPC (1.0.2) - 624 KB:**
Home page: *http://www.multiprecision.org/*
Download: *http://www.multiprecision.org/mpc/download/mpc-1.0.2.tar.gz*
MD5 sum: `68fadff3358fb3e7976c7a398a0af4c3`

- **MPFR (3.1.2) - 1,050 KB:**
Home page: *http://www.mpfr.org/*
Download: *http://www.mpfr.org/mpfr-3.1.2/mpfr-3.1.2.tar.xz*
MD5 sum: `e3d203d188b8fe60bb6578dd3152e05c`

- **Ncurses (5.9) - 2,764 KB:**
Home page: *http://www.gnu.org/software/ncurses*
Download: *http://ftp.gnu.org/gnu/ncurses/ncurses-5.9.tar.gz*
MD5 sum: `8cb9c412e5f2d96bc6f459aa8c6282a1`

- **Patch (2.7.1) - 668 KB:**
Home page: *http://savannah.gnu.org/projects/patch*
Download: *http://ftp.gnu.org/gnu/patch/patch-2.7.1.tar.xz*
MD5 sum: `e9ae5393426d3ad783a300a338c09b72`

- **Perl (5.20.0) - 13,770 KB:**
Home page: *http://www.perl.org*
Download: *http://www.cpan.org/src/5.0/perl-5.20.0.tar.bz2*
MD5 sum: `20cbecd4e9e880ee7a50a136c8b1484e`

- **Pkg-config-lite (0.28-1) - 384 KB:**
Home page: *http://sourceforge.net/projects/pkgconfiglite*
Download: *http://sourceforge.net/projects/pkgconfiglite/files/0.28-1/pkg-config-lite-0.28-1.tar.gz*
MD5 sum: `61f05feb6bab0a6bbfab4b6e3b2f44b6`

- **Procps-ng (3.3.9) - 548 KB:**
Home page: *http://sourceforge.net/projects/procps-ng*
Download: *http://sourceforge.net/projects/procps-ng/files/Production/procps-ng-3.3.9.tar.xz*
MD5 sum: `0980646fa25e0be58f7afb6b98f79d74`

- **Psmisc (22.21) - 458 KB:**
Home page: *http://psmisc.sourceforge.net*
Download: *http://downloads.sourceforge.net/psmisc/psmisc-22.21.tar.gz*
MD5 sum: `935c0fd6eb208288262b385fa656f1bf`

- **Readline (6.3) - 2,469 KB:**
Home page: *http://cnswww.cns.cwru.edu/php/chet/readline/rltop.html*
Download: *http://ftp.gnu.org/gnu/readline/readline-6.3.tar.gz*
MD5 sum: `33c8fb279e981274f485fd91da77e94a`

- **Rsyslog (6.4.2) - 2,519 KB:**
Home page: *http://www.rsyslog.com/*
Download: *http://www.rsyslog.com/files/download/rsyslog/rsyslog-6.4.2.tar.gz*
MD5 sum: `7de0124ec7d67ce2bfda0009ab1263ee`

- **Sed (4.2.2) - 1,036 KB:**
Home page: *http://www.gnu.org/software/sed*
Download: *http://ftp.gnu.org/gnu/sed/sed-4.2.2.tar.bz2*
MD5 sum: `7ffe1c7cdc3233e1e0c4b502df253974`

- **Shadow (4.2.1) - 1,595 KB:**
Home page: *http://pkg-shadow.alioth.debian.org*
Download: *http://pkg-shadow.alioth.debian.org/releases/shadow-4.2.1.tar.xz*
MD5 sum: `2bfafe7d4962682d31b5eba65dba4fc8`

- **Sysvinit (2.88dsf) - 104 KB:**
Home page: *http://savannah.nongnu.org/projects/sysvinit*
Download: *http://download.savannah.gnu.org/releases/sysvinit/sysvinit-2.88dsf.tar.bz2*
MD5 sum: `6eda8a97b86e0a6f59dabbf25202aa6f`

- **Tar (1.27.1) - 1,880 KB:**
Home page: *http://www.gnu.org/software/tar*
Download: *http://ftp.gnu.org/gnu/tar/tar-1.27.1.tar.xz*
MD5 sum: `e0382a4064e09a4943f3adeff1435978`

- **Tcl (8.6.1) - 8,756 KB:**
Home page: *http://www.tcl.tk*
Download: *http://downloads.sourceforge.net/tcl/tcl8.6.1-src.tar.gz*
MD5 sum: `aae4b701ee527c6e4e1a6f9c7399882e`

- **Texinfo (5.2) - 3,813 KB:**
Home page: *http://www.gnu.org/software/texinfo*
Download: *http://ftp.gnu.org/gnu/texinfo/texinfo-5.2.tar.xz*
MD5 sum: `cb489df8a7ee9d10a236197aefdb32c5`

- **Time Zone Data (2014d) - 221 KB:**
Home page: *http://www.iana.org/time-zones*
Download: *http://www.iana.org/time-zones/repository/releases/tzdata2014d.tar.gz*
MD5 sum: `299b86c0368ecfb321f15d5c408a1d9b`

- **Util-linux (2.24.2) - 3,587 KB:**
Download: *http://www.kernel.org/pub/linux/utils/util-linux/v2.24/util-linux-2.24.2.tar.xz*
MD5 sum: `3f191727a0d28f7204b755cf1b6ea0aa`

- **Vim (7.4) - 9,843 KB:**
Home page: *http://www.vim.org*
Download: *ftp://ftp.vim.org/pub/vim/unix/vim-7.4.tar.bz2*
MD5 sum: `607e135c559be642f210094ad023dc65`

- **XZ Utils (5.0.5) - 908 KB:**
Home page: *http://tukaani.org/xz/*
Download: *http://tukaani.org/xz/xz-5.0.5.tar.xz*
MD5 sum: `aa17280f4521dbeebed0fbd11cd7fa30`

- **Zlib (1.2.8) - 440 KB:**
Home page: *http://www.zlib.net*
Download: *http://zlib.net/zlib-1.2.8.tar.xz*
MD5 sum: `28f1205d8dd2001f26fec1e8c2cebe37`

Note

Zlib (1.2.8) may no longer be available at the listed location. The site administrators of the master download location occasionally remove older versions when new ones are released. An alternative download location that may have the correct version available is *http://cross-lfs.org/files/packages/3.0.0/SYSVINIT/*.

Total size of these packages: about 322 MB

3.3. Additional Packages for PowerPC

- **Hfsutils (3.2.6) - 204 KB:**
Home page: *http://www.mars.org/home/rob/proj/hfs*
Download: *ftp://ftp.mars.org/pub/hfs/hfsutils-3.2.6.tar.gz*
MD5 sum: `fa572afd6da969e25c1455f728750ec4`

- **Parted (3.1) - 1,492 KB:**
Home page: *http://www.gnu.org/software/parted*
Download: *http://ftp.gnu.org/gnu/parted/parted-3.1.tar.xz*
MD5 sum: `5d89d64d94bcfefa9ce8f59f4b81bdcb`

- **Powerpc-utils (1.1.3) - 28 KB:**
Home page: *http://packages.qa.debian.org/p/powerpc-utils.html*
Download: *http://ftp.debian.org/debian/pool/main/p/powerpc-utils/powerpc-utils_1.1.3.orig.tar.gz*
MD5 sum: `d879b109bb8f0d726304b60b147bff13`

- **Yaboot (1.3.17) - 220 KB:**
Home page: *http://yaboot.ozlabs.org*
Download: *http://yaboot.ozlabs.org/releases/yaboot-1.3.17.tar.gz*
MD5 sum: `f599f52d1887a86fd798252d2946f635`

Total size of these packages: about 1,944 KB

3.4. Needed Patches

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

- **Bash Branch Update Patch - 129 KB:**
Download: *http://patches.cross-lfs.org/3.0.0/SYSVINIT/bash-4.3-branch_update-5.patch*
MD5 sum: `712a693471a88bcece45fa566f8b6c57`
- **Bootscripts Cross-LFS Updates Patch - 16 KB:**
Download: *http://patches.cross-lfs.org/3.0.0/SYSVINIT/bootscripts-cross-lfs-3.0-20140710-tools_updates-2.patch*
MD5 sum: `460b0a4710dd74edea76b5398fd24b47`
- **Coreutils Uname Patch - 4.9 KB:**
Download: *http://patches.cross-lfs.org/3.0.0/SYSVINIT/coreutils-8.22-uname-1.patch*
MD5 sum: `6eeba217c88ec83b807e305e594fe13d`
- **Coreutils Noman Patch - 14 KB:**
Download: *http://patches.cross-lfs.org/3.0.0/SYSVINIT/coreutils-8.22-noman-1.patch*
MD5 sum: `6ed8f515391580e51f170a32af6fc7b9`
- **GCC Branch Update Patch - 3.5 KB:**
Download: *http://patches.cross-lfs.org/3.0.0/SYSVINIT/gcc-4.8.3-branch_update-1.patch*
MD5 sum: `ba8abbb0696f8e0d75eb26ae7c9ad219`
- **Iana-Etc Protocol and Port Numbers Update - 282 KB:**
Download: *http://patches.cross-lfs.org/3.0.0/SYSVINIT/iana-etc-2.30-numbers_update-20140202-2.patch.xz*
MD5 sum: `b0e7051fef0b3ba064209a5f3d23bd2a`
- **IPUtils Fixes Patch - 158 KB:**
Download: *http://patches.cross-lfs.org/3.0.0/SYSVINIT/iputils-s20121221-fixes-2.patch*
MD5 sum: `c2344acdd81607685886f617da7d66e0`
- **Linux Sublevel Patch - 511 KB:**
Download: *http://patches.cross-lfs.org/3.0.0/SYSVINIT/patch-3.14.21.xz*
MD5 sum: `25debf3b5652cdd94df176cd4e36a9ed`
- **MPFR Fixes Patch - 55 KB:**
Download: *http://patches.cross-lfs.org/3.0.0/SYSVINIT/mpfr-3.1.2-fixes-4.patch*
MD5 sum: `b6c1c0dcbf7661298037eeb346a8669c`
- **Ncurses Bash Patch - .743 KB:**
Download: *http://patches.cross-lfs.org/3.0.0/SYSVINIT/ncurses-5.9-bash_fix-1.patch*
MD5 sum: `c6f7f2ab0ebaf7721ebeb266641352db`
- **Ncurses Branch Update Patch - 2,492 KB:**
Download: *http://patches.cross-lfs.org/3.0.0/SYSVINIT/ncurses-5.9-branch_update-4.patch*
MD5 sum: `c2b2dc2d31b02c218359e6218f12a72c`
- **Readline Branch Update - 5.733 KB:**
Download: *http://patches.cross-lfs.org/3.0.0/SYSVINIT/readline-6.3-branch_update-4.patch*
MD5 sum: `bb8d37fd00abc74a81563ebfcf64af4c`

- **Sysvinit Tools Updates Patch - 2.339 KB:**
Download: *http://patches.cross-lfs.org/3.0.0/SYSVINIT/sysvinit-2.88dsf-tools_updates-1.patch*
MD5 sum: `c3f6981c46868b68bfd58921570ea51f`
- **Tar Man Page Patch - 7.924 KB:**
Download: *http://patches.cross-lfs.org/3.0.0/SYSVINIT/tar-1.27.1-manpage-1.patch*
MD5 sum: `68c86c67e67a5c074872a293818f361d`
- **Vim Branch Update Patch - 3,547 KB:**
Download: *http://patches.cross-lfs.org/3.0.0/SYSVINIT/vim-7.4-branch_update-7.patch*
MD5 sum: `3c250be2ca0cf3e539a8f18c46c36eac`

Total size of these patches: about 7 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/3.0.0/SYSVINIT/* and acquire any additional patches to suit the system needs.

3.5. Additional Patches for PowerPC

- **GCC Specs Patch - 20 KB:**
Download: *http://patches.cross-lfs.org/3.0.0/SYSVINIT/gcc-4.8.3-specs-1.patch*
MD5 sum: `118eaea7813c24cb3fcfdafa0297abba`
- **HFS Utils Fixes Patch - 1.1 KB:**
Download: *http://patches.cross-lfs.org/3.0.0/SYSVINIT/hfsutils-3.2.6-fixes-1.patch*
MD5 sum: `8519f11aada2f393609d529621a9f1b1`
- **Powerpc-utils Fixes Patch - 22 KB:**
Download: *http://patches.cross-lfs.org/3.0.0/SYSVINIT/powerpc-utils_1.1.3-fixes-2.patch*
MD5 sum: `d2776b1a4977c5711037b8f1402f792a`
- **Yaboot Ofpath_Path_Prefix Patch - .830 KB:**
Download: *http://patches.cross-lfs.org/3.0.0/SYSVINIT/yaboot-1.3.17-ofpath_path_prefix-1.patch*
MD5 sum: `3faf70e0cb4e4f62a1e8815c3452ab38`
- **Yaboot Parted Patch - 1.245 KB:**
Download: *http://patches.cross-lfs.org/3.0.0/SYSVINIT/yaboot-1.3.17-parted-1.patch*
MD5 sum: `af05081b2056f0070b8b057acc32fea5`
- **Yaboot Stubfuncs Patch - 4.2 KB:**
Download: *http://patches.cross-lfs.org/3.0.0/SYSVINIT/yaboot-1.3.17-stubfuncs-1.patch*
MD5 sum: `b5cc91f9904383c24848040bfe6f11ae`

Total size of these patches: about 49.375 KB

Chapter 4. Final Preparations

4.1. Introduction

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

4.2. Creating the ${CLFS}/tools Directory

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

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

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

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

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

Note

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

The created symlink enables the toolchain to be compiled so that it always refers to `/tools`, meaning that the compiler, assembler, and linker will work. This will provide a common place for our temporary tools system.

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

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

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

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

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

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

4.4. Adding the CLFS User

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

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

The meaning of the command line options:

`-s /bin/bash`

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

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

`-g clfs`

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

`-d /home/clfs`

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

`clfs`

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

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

```
passwd clfs
```

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

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

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

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

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

```
su - clfs
```

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

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

4.5. Setting Up the Environment

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

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

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

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

```
cat > ~/.bashrc << "EOF"
set +h
umask 022
CLFS=/mnt/clfs
LC_ALL=POSIX
PATH=/cross-tools/bin:/bin:/usr/bin
export CLFS LC_ALL PATH
unset CFLAGS CXXFLAGS
EOF
```

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

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

The `CLFS` variable should be set to the chosen mount point.

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

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

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

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

```
source ~/.bash_profile
```

4.6. Build Variables

Setting Host and Target

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

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

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

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

Copy settings to Environment

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

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

4.7. About the Test Suites

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

Part III. Make the Cross-Compile Tools

Chapter 5. Constructing Cross-Compile Tools

5.1. Introduction

This chapter shows you how to create cross platform tools.

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

5.1.1. Common Notes

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

Several of the packages are patched before compilation, but only when the patch is needed to circumvent a problem. A patch is often needed in both this and the next chapters, but sometimes in only one or the other. Therefore, do not be concerned if instructions for a downloaded patch seem to be missing. Warning messages about *offset* or *fuzz* may also be encountered when applying a patch. Do not worry about these warnings, as the patch was still successfully applied.

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

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

5.2. File-5.19

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

5.2.1. Installation of File

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

Prepare File for compilation:

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

The meaning of the configure options:

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

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

Compile the package:

```
make
```

Install the package:

```
make install
```

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

5.3. Linux-3.14.21 Headers

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

5.3.1. Installation of Linux Headers

> **Note**
>
> For this step you will need to unpack the kernel tarball (`linux-3.14.tar.xz`) and **cd** into its source directory before entering the commands on this page.

Apply the latest Linux sublevel patch:

```
xzcat ../patch-3.14.21.xz | patch -Np1 -i -
```

Install the kernel header files:

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

The meaning of the make commands:

`make mrproper`
: Ensures that the kernel source dir is clean.

`make ARCH=powerpc headers_check`
: Sanitizes the raw kernel headers so that they can be used by userspace programs.

`make ARCH=powerpc INSTALL_HDR_PATH=/tools headers_install`
: This will install the kernel headers into `/tools/include`.

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

5.4. M4-1.4.17

The M4 package contains a macro processor.

5.4.1. Installation of M4

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

Prepare M4 for compilation:

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

Compile the package:

```
make
```

Install the package:

```
make install
```

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

5.5. Ncurses-5.9

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

5.5.1. Installation of Ncurses

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

The following patch fixes an issue with some Bash versions:

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

Prepare Ncurses for compilation:

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

The meaning of the new configure options:

`--without-debug`
Tells Ncurses to build without debugging information.

`--without-shared`
This prevents Ncurses from building its shared libraries, which are not needed at this time.

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

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

Install **tic** with the following command:

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

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

5.6. Pkg-config-lite-0.28-1

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

5.6.1. Installation of Pkg-config-lite

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

Prepare Pkg-config-lite for compilation:

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

The meaning of the new configure option:

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

`--with-pc-path`
: This sets the default PKG_CONFIG_PATH to `/tools/lib/pkgconfig` and `/tools/share/pkgconfig`.

Compile the package:

```
make
```

Install the package:

```
make install
```

Details on this package are located in Section 10.21.2, “Contents of Pkg-config-lite.”

5.7. GMP-6.0.0

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

5.7.1. Installation of GMP

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

Note

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

Prepare GMP for compilation:

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

The meaning of the new configure option:

`--enable-cxx`
: This tells GMP to enable C++ support.

Compile the package:

```
make
```

Install the package:

```
make install
```

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

5.8. MPFR-3.1.2

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

5.8.1. Installation of MPFR

Apply a patch with upstream fixes:

```
patch -Np1 -i ../mpfr-3.1.2-fixes-4.patch
```

Prepare MPFR for compilation:

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

The meaning of the new configure options:

LDFLAGS="-Wl,-rpath,/cross-tools/lib"
: This tells **configure** to search in `/cross-tools` for libraries.

--with-gmp=/cross-tools
: This tells **configure** where to find GMP.

Compile the package:

```
make
```

Install the package:

```
make install
```

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

5.9. MPC-1.0.2

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

5.9.1. Installation of MPC

Prepare MPC for compilation:

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

Compile the package:

```
make
```

Install the package:

```
make install
```

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

5.10. ISL-0.12.2

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

5.10.1. Installation of ISL

We will install ISL and CLooG to enable extra functionality for GCC. They are not strictly required, but GCC can link to them to enable its new loop generation feature called Graphite.

Prepare ISL for compilation:

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

Compile the package:

```
make
```

Install the package:

```
make install
```

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

5.11. CLooG-0.18.2

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

5.11.1. Installation of CLooG

Prepare CLooG for compilation:

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

Apply a **sed** which prevents the attempted installation of an invalid file:

```
cp -v Makefile{,.orig}
sed '/cmake/d' Makefile.orig > Makefile
```

Compile the package:

```
make
```

Install the package:

```
make install
```

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

5.12. Cross Binutils-2.24

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

5.12.1. Installation of Cross Binutils

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

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

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

Prepare Binutils for compilation:

```
AR=ar AS=as ../binutils-2.24/configure \
    --prefix=/cross-tools --host=${CLFS_HOST} --target=${CLFS_TARGET} \
    --with-sysroot=${CLFS} --with-lib-path=/tools/lib --disable-nls \
    --disable-static --disable-multilib --disable-werror
```

The meaning of the new configure options:

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

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

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

--with-sysroot=${CLFS}
: Tells configure to build a linker that uses ${CLFS} as its root directory for its search paths.

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

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

--disable-multilib
: This option disables the building of a multilib capable Binutils.

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

Compile the package:

```
make
```

Install the package:

```
make install
```

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

5.13. Cross GCC-4.8.3 - Static

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

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

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

The following patch contains a number of updates to the 4.8.3 branch by the GCC developers:

```
patch -Np1 -i ../gcc-4.8.3-branch_update-1.patch
```

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

```
patch -Np1 -i ../gcc-4.8.3-specs-1.patch
```

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

```
echo -en '\n#undef STANDARD_STARTFILE_PREFIX_1\n#define STANDARD_STARTFILE_PREFIX
echo -en '\n#undef STANDARD_STARTFILE_PREFIX_2\n#define STANDARD_STARTFILE_PREFIX
```

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

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

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

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

Prepare GCC for compilation:

```
AR=ar LDFLAGS="-Wl,-rpath,/cross-tools/lib" \
    ../gcc-4.8.3/configure --prefix=/cross-tools \
    --build=${CLFS_HOST} --host=${CLFS_HOST} --target=${CLFS_TARGET} \
    --with-sysroot=${CLFS} --with-local-prefix=/tools \
    --with-native-system-header-dir=/tools/include --disable-nls \
    --disable-shared --with-mpfr=/cross-tools --with-gmp=/cross-tools \
    --with-isl=/cross-tools --with-cloog=/cross-tools --with-mpc=/cross-tools \
    --without-headers --with-newlib --disable-decimal-float --disable-libgomp \
    --disable-libmudflap --disable-libssp --disable-libatomic --disable-libitm \
    --disable-libsanitizer --disable-libquadmath --disable-threads \
    --disable-multilib --disable-target-zlib --with-system-zlib \
    --enable-languages=c --enable-checking=release
```

The meaning of the new configure options:

`--build=${CLFS_HOST}`

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

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

--with-native-system-headers-dir=/tools/include
: This switch ensures that GCC will search for the system headers in `/tools/include` and that host system headers will not be searched.

--disable-shared
: This tells GCC not to create a shared library.

--without-headers
: Disables GCC from using the target's Libc when cross compiling.

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

--disable-decimal-float
: Disables support for the C decimal floating point extension.

*--disable-lib**
: These options prevent GCC from building a number of libraries that are not needed at this time.

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

--disable-target-zlib
: This tells GCC not to build the copy of Zlib in its source tree.

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

--enable-languages=c
: This option ensures that only the C compiler is built.

--enable-checking=release
: This option selects the complexity of the internal consistency checks and adds error checking within the compiler.

Continue with compiling the package:

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

The meaning of the new make options:

all-gcc all-target-libgcc
: Compiles only the parts of GCC that are needed at this time, rather than the full package.

Install the package:

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

Details on this package are located in Section 10.19.2, “Contents of GCC.”

5.14. Glibc-2.19

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.

5.14.1. 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.

Apply the following **sed** so the **tzselect** script works properly:

```
cp -v timezone/Makefile{,.orig}
sed 's/\\$$(pwd)/`pwd`/' timezone/Makefile.orig > timezone/Makefile
```

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

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

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

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

Prepare Glibc for compilation:

```
BUILD_CC="gcc" CC="${CLFS_TARGET}-gcc" \
    AR="${CLFS_TARGET}-ar" RANLIB="${CLFS_TARGET}-ranlib" \
    ../glibc-2.19/configure --prefix=/tools \
    --host=${CLFS_TARGET} --build=${CLFS_HOST} \
    --disable-profile --enable-kernel=2.6.32 \
    --with-binutils=/cross-tools/bin --with-headers=/tools/include \
    --enable-obsolete-rpc --cache-file=config.cache
```

The meaning of the new configure options:

`BUILD_CC="gcc"`
: This sets Glibc to use the current compiler on our system. This is used to create the tools Glibc uses during its build.

`CC="${CLFS_TARGET}-gcc"`
: This forces Glibc to use the GCC compiler that we made for our target architecture.

`AR="${CLFS_TARGET}-ar"`
: This forces Glibc to use the **ar** utility we made for our target architecture.

`RANLIB="${CLFS_TARGET}-ranlib"`
: This forces Glibc to use the **ranlib** utility we made for our target architecture.

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

--enable-kernel=2.6.32
This tells Glibc to compile the library with support for 2.6.32 and later Linux kernels.

--with-binutils=/cross-tools/bin
This tells Glibc to use the Binutils that are specific to our target architecture.

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

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

--cache-file=config.cache
This tells Glibc to utilize a premade cache file.

During this stage the following warning might appear:

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

The missing or incompatible **msgfmt** program is generally harmless. This **msgfmt** program is part of the Gettext package which the host distribution should provide. You might also see a similar (also harmless) message about missing **autoconf**.

Compile the package:

```
make
```

Install the package:

```
make install
```

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

5.15. Cross GCC-4.8.3 - Final

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

5.15.1. Installation of GCC Cross Compiler

The following patch contains a number of updates to the 4.8.3 branch by the GCC developers:

```
patch -Np1 -i ../gcc-4.8.3-branch_update-1.patch
```

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

```
patch -Np1 -i ../gcc-4.8.3-specs-1.patch
```

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

```
echo -en '\n#undef STANDARD_STARTFILE_PREFIX_1\n#define STANDARD_STARTFILE_PREFIX
echo -en '\n#undef STANDARD_STARTFILE_PREFIX_2\n#define STANDARD_STARTFILE_PREFIX
```

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

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

Prepare GCC for compilation:

```
AR=ar LDFLAGS="-Wl,-rpath,/cross-tools/lib" \
    ../gcc-4.8.3/configure --prefix=/cross-tools \
    --build=${CLFS_HOST} --target=${CLFS_TARGET} --host=${CLFS_HOST} \
    --with-sysroot=${CLFS} --with-local-prefix=/tools \
    --with-native-system-header-dir=/tools/include --disable-nls \
    --disable-static --enable-languages=c,c++ --enable-__cxa_atexit \
    --enable-threads=posix --disable-multilib \
    --with-mpc=/cross-tools --with-mpfr=/cross-tools --with-gmp=/cross-tools \
    --with-cloog=/cross-tools --with-isl=/cross-tools --with-system-zlib \
    --enable-checking=release --enable-libstdcxx-time
```

The meaning of the new configure options:

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

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

--enable-threads=posix
: This enables C++ exception handling for multi-threaded code.

--enable-libstdcxx-time
: This enables link-time checks for the availability of clock_gettime clocks, and nanosleep and sched_yield functions, in the C library.

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 10.19.2, “Contents of GCC.”

Part IV. Building the Basic Tools

Chapter 6. Constructing a Temporary System

6.1. Introduction

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

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

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

```
echo ${CLFS}
```

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

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

```
configure: WARNING: result yes guessed because of cross compilation
configure: WARNING: cannot check WCONTINUED if cross compiling -- defaulting to
```

6.2. Build Variables

Setup target-specific variables for the compiler and linkers:

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

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

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

6.3. GMP-6.0.0

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

6.3.1. Installation of GMP

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

Prepare GMP for compilation:

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

The meaning of the new configure option:

CC_FOR_BUILD=gcc

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

Compile the package:

```
make
```

Install the package:

```
make install
```

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

6.4. MPFR-3.1.2

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

6.4.1. Installation of MPFR

Apply a patch with upstream fixes:

```
patch -Np1 -i ../mpfr-3.1.2-fixes-4.patch
```

Prepare MPFR for compilation:

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

Compile the package:

```
make
```

Install the package:

```
make install
```

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

6.5. MPC-1.0.2

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

6.5.1. Installation of MPC

Prepare MPC for compilation:

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

Compile the package:

```
make
```

Install the package:

```
make install
```

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

6.6. ISL-0.12.2

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

6.6.1. Installation of ISL

Prepare ISL for compilation:

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

Compile the package:

```
make
```

Install the package:

```
make install
```

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

6.7. CLooG-0.18.2

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

6.7.1. Installation of CLooG

Prepare CLooG for compilation:

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

The meaning of the new configure option:

--with-isl=system
: This ensures that CLooG will use the version of Isl that was just installed into `/tools`, rather than the copy in its own source tree.

Apply a **sed** which prevents the attempted installation of an invalid file:

```
cp -v Makefile{,.orig}
sed '/cmake/d' Makefile.orig > Makefile
```

Compile the package:

```
make
```

Install the package:

```
make install
```

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

6.8. Zlib-1.2.8

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

6.8.1. Installation of Zlib

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

Prepare Zlib for compilation:

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

Compile the package:

```
make
```

Install the package:

```
make install
```

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

6.9. Binutils-2.24

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

6.9.1. Installation of Binutils

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

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

Prepare Binutils for compilation:

```
../binutils-2.24/configure --prefix=/tools \
    --build=${CLFS_HOST} --host=${CLFS_TARGET} --target=${CLFS_TARGET} \
    --with-lib-path=/tools/lib --disable-nls --enable-shared \
    --disable-multilib
```

The meaning of the new configure option:

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

Compile the package:

```
make
```

Install the package:

```
make install
```

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

6.10. GCC-4.8.3

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

6.10.1. Installation of GCC

The following patch contains a number of updates to the 4.8.3 branch by the GCC developers:

```
patch -Np1 -i ../gcc-4.8.3-branch_update-1.patch
```

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

```
patch -Np1 -i ../gcc-4.8.3-specs-1.patch
```

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

```
echo -en '\n#undef STANDARD_STARTFILE_PREFIX_1\n#define STANDARD_STARTFILE_PREFIX
echo -en '\n#undef STANDARD_STARTFILE_PREFIX_2\n#define STANDARD_STARTFILE_PREFIX
```

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

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

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

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

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

Prepare GCC for compilation:

```
../gcc-4.8.3/configure --prefix=/tools \
    --build=${CLFS_HOST} --host=${CLFS_TARGET} --target=${CLFS_TARGET} \
    --with-local-prefix=/tools --disable-multilib --disable-nls \
    --enable-languages=c,c++ --disable-libstdcxx-pch \
    --with-system-zlib --with-native-system-header-dir=/tools/include \
    --disable-libssp --enable-checking=release --enable-libstdcxx-time
```

The meaning of the new configure option:

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

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

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

Compile the package:

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

Install the package:

```
make install
```

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

```
cp -v ../gcc-4.8.3/include/libiberty.h /tools/include
```

Details on this package are located in Section 10.19.2, “Contents of GCC.”

6.11. Ncurses-5.9

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

6.11.1. Installation of Ncurses

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

The following patch fixes an issue with some Bash versions:

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

Prepare Ncurses for compilation:

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

The meaning of the new configure options:

--with-shared
: This tells Ncurses to create a shared library.

--without-debug
: This tells Ncurses not to build with debug information.

--without-ada
: This ensures that Ncurses does not build support for the Ada compiler which may be present on the host but will not be available when building the final system.

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

--with-build-cc=gcc
: This tells Ncurses which compiler to use to build native tools when cross-compiling.

Compile the package:

```
make
```

Install the package:

```
make install
```

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

6.12. Bash-4.3

The Bash package contains the Bourne-Again SHell.

6.12.1. Installation of Bash

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

```
patch -Np1 -i ../bash-4.3-branch_update-5.patch
```

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

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

Prepare Bash for compilation:

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

The meaning of the new configure option:

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

Compile the package:

```
make
```

Install the package:

```
make install
```

Details on this package are located in Section 10.37.2, “Contents of Bash.”

6.13. Bzip2-1.0.6

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

6.13.1. Installation of Bzip2

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

```
cp -v Makefile{,.orig}
sed -e 's@^\(all:.*\) test@\1@g' Makefile.orig > Makefile
```

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

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

Install the package:

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

Details on this package are located in Section 10.31.2, “Contents of Bzip2.”

6.14. Check-0.9.13

The Check package is a unit testing framework for C.

6.14.1. Installation of Check

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

Prepare Check for compilation:

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

Build the package:

```
make
```

Install the package:

```
make install
```

6.14.2. Contents of Check

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

Short Descriptions

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

6.15. Coreutils-8.22

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

6.15.1. Installation of Coreutils

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

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

Apply a patch to prevent Coreutils from generating manpages :

```
patch -Np1 -i ../coreutils-8.22-noman-1.patch
```

Prepare Coreutils for compilation:

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

The meaning of the new configure option:

`--enable-install-program=hostname`
: Tells Coreutils to install **hostname**, which is needed for the Perl test suite.

Compile the package:

```
make
```

Install the package:

```
make install
```

Details on this package are located in Section 10.27.2, “Contents of Coreutils.”

6.16. Diffutils-3.3

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

6.16.1. Installation of Diffutils

Prepare Diffutils for compilation:

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

Compile the package:

```
make
```

Install the package:

```
make install
```

Details on this package are located in Section 10.39.2, "Contents of Diffutils."

6.17. File-5.19

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

6.17.1. Installation of File

Prepare File for compilation:

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

Compile the package:

```
make
```

Install the package:

```
make install
```

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

6.18. Findutils-4.4.2

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

6.18.1. Installation of Findutils

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

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

Prepare Findutils for compilation:

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

Compile the package:

```
make
```

Install the package:

```
make install
```

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

6.19. Gawk-4.1.1

The Gawk package contains programs for manipulating text files.

6.19.1. Installation of Gawk

Prepare Gawk for compilation:

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

Compile the package:

```
make
```

Install the package:

```
make install
```

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

6.20. Gettext-0.19.1

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

6.20.1. Installation of Gettext

Many packages' installation procedures use the **msgfmt** program for i18n support, so we will compile and install it into `/tools`.

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 even when we do. The following will fix possible compilation errors because of this assumption:

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

Prepare Gettext for compilation:

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

Compile the required program and support library:

```
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 10.43.2, “Contents of Gettext.”

6.21. Grep-2.19

The Grep package contains programs for searching through files.

6.21.1. Installation of Grep

Prepare Grep for compilation:

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

The meaning of the new configure option:

--without-included-regex

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

Compile the package:

```
make
```

Install the package:

```
make install
```

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

6.22. Gzip-1.6

The Gzip package contains programs for compressing and decompressing files.

6.22.1. Installation of Gzip

Prepare Gzip for compilation:

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

Compile the package:

```
make
```

Install the package:

```
make install
```

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

6.23. Make-4.0

The Make package contains a program for compiling packages.

6.23.1. Installation of Make

Prepare Make for compilation:

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

Compile the package:

```
make
```

Install the package:

```
make install
```

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

6.24. Patch-2.7.1

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

6.24.1. Installation of Patch

Prepare Patch for compilation:

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

Compile the package:

```
make
```

Install the package:

```
make install
```

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

6.25. Sed-4.2.2

The Sed package contains a stream editor.

6.25.1. Installation of Sed

Prepare Sed for compilation:

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

Compile the package:

```
make
```

Install the package:

```
make install
```

Details on this package are located in Section 10.20.2, “Contents of Sed.”

6.26. Tar-1.27.1

The Tar package contains an archiving program.

6.26.1. Installation of Tar

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

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

Prepare Tar for compilation:

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

Compile the package:

```
make
```

Install the package:

```
make install
```

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

6.27. Texinfo-5.2

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

6.27.1. Installation of Texinfo

Prepare Texinfo for compilation:

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

The meaning of the new configure option:

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

Compile the package:

```
make
```

Install the package:

```
make install
```

Details on this package are located in Section 10.62.2, “Contents of Texinfo.”

6.28. Util-linux-2.24.2

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

6.28.1. Installation of Util-linux

Prepare Util-linux for compilation:

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

The meaning of the new configure option:

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

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

Compile the package:

```
make
```

Install the package:

```
make install
```

Details on this package are located in Section 10.24.3, “Contents of Util-linux.”

6.29. Vim-7.4

The Vim package contains a powerful text editor.

6.29.1. Installation of VIM

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

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

```
patch -Np1 -i ../vim-7.4-branch_update-7.patch
```

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

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

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

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

Prepare Vim for compilation:

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

The meaning of the new configure options:

--enable-gui=no --disable-gtktest --disable-xim --disable-gpm --without-x --disable-netbeans
: These options prevent Vim from trying to link to libraries that might be on the host but won't exist inside the temporary build environment.

--with-tlib=ncurses
: Tells Vim to use Ncurses as its terminal library.

Compile the package:

```
make
```

Install the package:

```
make install
```

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

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

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

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

set nocompatible
set backspace=2
set ruler
syntax on

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

Details on this package are located in Section 10.64.3, “Contents of Vim.”

6.30. XZ Utils-5.0.5

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

6.30.1. Installation of XZ Utils

Prepare XZ Utils for compilation:

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

Compile the package:

```
make
```

Install the package:

```
make install
```

Details on this package are located in Section 10.53.2, “Contents of XZ Utils.”

6.31. To Boot or to Chroot?

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

The **chroot** (change root) program is used to enter a virtual environment and start a new shell whose root directory will be set to the CLFS partition. This is very similar to rebooting and instructing the kernel to mount the CLFS partition as the root partition. The major advantage is that “chrooting” allows the builder to continue using the host while CLFS is being built. While waiting for package compilation to complete, a user can switch to a different virtual console (VC) or X desktop and continue using the computer as normal.

The main downside to chrooting is that you are more limited in when you can use it - booting will always work for any CLFS build, but the chroot method can only be used when you are building on the same architecture. For example, if you are building on, and for, an x86 system, you can simply chroot. Booting is required when you are compiling for a different architecture, such as building a PowerPC system from an x86. The rule of thumb here is that if the architectures match and you are running the same series kernel (specifically, a 2.6.32 or newer Linux kernel) you can just chroot. If you aren't running the same series kernel, or are wanting to run a different ABI, you will need to use the boot option.

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

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

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

For the boot method, follow If You Are Going to Boot.

For the chroot method, follow If You Are Going to Chroot.

Chapter 7. If You Are Going to Boot

7.1. Introduction

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

7.2. Bc-1.06.95

The Bc package contains an arbitrary precision numeric processing language.

7.2.1. Installation of Bc

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

Prepare Bc for compilation:

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

The meaning of the configure option:

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

Compile the package:

```
make
```

Install the package:

```
make install
```

Details on this package are located in Section 10.38.2, “Contents of Bc.”

7.3. Bootscripts for CLFS 3.0-20140710

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

7.3.1. Installation of Bootscripts

Apply a patch which modifies the scripts for use in `/tools`:

```
patch -Np1 -i ../bootscripts-cross-lfs-3.0-20140710-tools_updates-2.patch
```

Install the package:

```
make DESTDIR=/tools 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 `/tools/etc/sysconfig/clock` file (which tells the **hwclock** program which timezone the user is in). There is no way to detect whether or not the hardware clock is set to UTC, so this needs to be configured manually.

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

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

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

UTC=1

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

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

7.4. E2fsprogs-1.42.9

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

7.4.1. Installation of E2fsprogs

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

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

Prepare E2fsprogs for compilation:

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

The meaning of the configure options:

--enable-elf-shlibs
: This creates the shared libraries which some programs in this package use.

*--disable-**
: This prevents E2fsprogs from building and installing the `libuuid` and `libblkid` libraries, the `uuidd` daemon, and the **fsck** wrapper, as Util-Linux installed all of them earlier.

Compile the package:

```
make
```

Install the binaries, documentation and shared libraries:

```
make install
```

Install the static libraries and headers:

```
make install-libs
```

Details on this package are located in Section 10.26.2, “Contents of E2fsprogs.”

7.5. Kmod-18

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

7.5.1. Installation of Kmod

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

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

Prepare Kmod for compilation:

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

The meaning of the new configure options:

`--with-zlib --with-xz`
: These allow the Kmod package to handle zlib and XZ compressed kernel modules.

Compile the package:

```
make
```

Install the package:

```
make install
```

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

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

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

7.6. Shadow-4.2.1

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

7.6.1. Installation of Shadow

Run the following **sed** command to disable the installation of the **groups** and **nologin** programs, as better versions of these programs are provided by other packages, and prevent Shadow from setting the suid bit on its installed programs:

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

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

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

Prepare Shadow for compilation:

```
./configure --prefix=/tools \
    --build=${CLFS_HOST} --host=${CLFS_TARGET} --cache-file=config.cache \
    --enable-subordinate-ids=no
```

The meaning of the configure options:

--enable-subordinate-ids=no
: Disable subordinate ids option during cross-compile.

Append to `config.h` since a test program will not be ran when cross-compiling:

```
echo "#define ENABLE_SUBUIDS 1" >> config.h
```

Compile the package:

```
make
```

Install the package:

```
make install
```

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

7.7. Sysvinit-2.88dsf

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

7.7.1. Installation of Sysvinit

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

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

Compile the package:

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

Install the package:

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

7.7.2. Configuring Sysvinit

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

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

id:3:initdefault:

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

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

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

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

EOF
```

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

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

EOF
```

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

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

EOF
```

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

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

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

Details on this package are located in Section 10.60.3, “Contents of Sysvinit.”

7.8. Eudev-1.7

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

7.8.1. Installation of Eudev

Prepare Eudev for compilation:

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

The meaning of the new configure options:

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

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

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

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

--enable-libkmod

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

Compile the package:

```
make
```

Install the package:

```
make install
```

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

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

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

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

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

7.9. Linux-3.14.21

The Linux package contains the Linux kernel.

7.9.1. Installation of the kernel

> **Warning**
>
> Here a temporary cross-compiled kernel will be built. When configuring it, select the minimal amount of options required to boot the target machine and build the final system. I.e., no support for sound, printers, etc. will be needed.
>
> Also, try to avoid the use of modules if possible, and don't use the resulting kernel image for production systems.

Building the kernel involves a few steps—configuration, compilation, and installation. Read the `README` file in the kernel source tree for alternative methods to the way this book configures the kernel.

Apply the latest Linux sublevel patch:

```
xzcat ../patch-3.14.21.xz | patch -Np1 -i -
```

Prepare for compilation by running the following command:

```
make mrproper
```

This ensures that the kernel tree is absolutely clean. The kernel team recommends that this command be issued prior to each kernel compilation. Do not rely on the source tree being clean after un-tarring.

Configure the kernel via a menu-driven interface:

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

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

Compile the kernel image and modules:

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

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

Install the modules, if the kernel configuration uses them:

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

Install the firmware, if the kernel configuration uses them:

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

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

Issue the following commands to install the kernel:

```
mkdir -pv /tools/boot
cp -v vmlinux /tools/boot/clfskernel-3.14.21
```

`System.map` is a symbol file for the kernel. It maps the function entry points of every function in the kernel API, as well as the addresses of the kernel data structures for the running kernel. Issue the following command to install the map file:

```
cp -v System.map /tools/boot/System.map-3.14.21
```

The kernel configuration file `.config` produced by the **make menuconfig** step above contains all the configuration selections for the kernel that was just compiled. It is a good idea to keep this file for future reference:

```
cp -v .config /tools/boot/config-3.14.21
```

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

7.10. Hfsutils-3.2.6

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

7.10.1. Installation of Hfsutils

If you have created, or will create, the ext2 filesystem on your Mac using ext2fsx you can jump ahead to Section 7.12, “Yaboot-1.3.17.”. The next three packages are for people who cannot do that.

Apply the following patch to add a missing `errno.h` include and allow HFSutils to recognize devices larger than 2GB:

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

Prepare Hfsutils for compilation:

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

Compile the package:

```
make
```

Install the package:

```
make install
```

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

7.11. Powerpc-Utils_1.1.3

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

7.11.1. Installation of Powerpc-Utils

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

```
patch -Np1 -i ../powerpc-utils_1.1.3-fixes-2.patch
```

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

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

Install the program:

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

Details on this package are located in Section 10.67.2, “Contents of Powerpc-Utils.”

7.12. Yaboot-1.3.17

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

7.12.1. Installation of Yaboot

The following patch adds stub functions for newer e2fsprogs releases:

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

The following patch adds Parted support to yabootconfig:

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

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

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

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

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

Compile the package:

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

Install the package:

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

Details on this package are located in Section 10.68.2, “Contents of Yaboot.”

7.13. Creating Directories

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

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

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

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

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

7.13.1. 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.

7.14. Creating Essential Symlinks

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

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

The purpose of each link:

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

/bin/cat
: This pathname is hard-coded into Glibc's configure script.

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

/bin/grep
: This to avoid a hard-coded `/tools` reference in Libtool.

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

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

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

/usr/bin/file
: Binutils' **configure** scripts specify this command location.

/usr/lib/libgcc_s.so{,.1}
: Glibc needs this for the pthreads library to work.

/usr/lib/libstdc++{,.6}
: This is needed by several tests in Glibc's test suite, as well as for C++ support in GMP.

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

/bin/sh
: Many shell scripts hard-code `/bin/sh`.

/sbin/init

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

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

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

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

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

7.15. Populating /dev

7.15.1. Creating Initial Device Nodes

When the kernel boots the system, it requires the presence of a few device nodes, in particular the `console` and `null` devices. The device nodes will be created on the hard disk so that they are available before **udev** has been started, and additionally when Linux is started in single user mode (hence the restrictive permissions on `console`). Create these by running the following commands:

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

7.16. 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
bin:x:1:1:/bin:/bin/false
daemon:x:2:6:/sbin:/bin/false
nobody:x:65534:65533:Unprivileged User:/dev/null:/bin/false
EOF
```

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

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

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

Was used for programs that performed administrative tasks.

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

Used by programs for printing

`mail:x:30:30:mail:/var/mail:/bin/false`
: Often used by email programs

`messagebus:x:27:27:D-Bus Message Daemon User:/dev/null:/bin/false`
: User for D-Bus

`news:x:31:31:news:/var/spool/news:/bin/false`
: Often used for network news servers

`operator:x:50:0:operator:/root:/bin/bash`
: Often used to allow system operators to access the system

`postmaster:x:51:30:postmaster:/var/spool/mail:/bin/false`
: Generally used as an account that receives all the information of troubles with the mail server

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

```
cat > ${CLFS}/etc/group << "EOF"
root:x:0:
bin:x:1:
sys:x:2:
kmem:x:3:
tty:x:5:
tape:x:4:
daemon:x:6:
floppy:x:7:
disk:x:8:
lp:x:9:
dialout:x:10:
audio:x:11:
video:x:12:
utmp:x:13:
usb:x:14:
cdrom:x:15:
adm:x:16:
mail:x:30:
wheel:x:39:
nogroup:x:65533:
EOF
```

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

`console:x:17:`
: This group has direct access to the console

`cdrw:x:18:`
: This group is allowed to use the CDRW drive

`messagebus:x:27:`
: User for D-Bus

`news:x:31:news`
: Used by Network News Servers

`users:x:1000:`
: The default GID used by shadow for new users

`nobody:x:65533:`
: This is used by NFS

The created groups are not part of any standard—they are groups decided on in part by the requirements of the Systemd configuration in the final system, and in part by common convention employed by a number of existing Linux distributions. The Linux Standard Base (LSB, available at *http://www.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/log/{btmp,lastlog,wtmp}
chgrp -v 13 ${CLFS}/var/log/lastlog
chmod -v 664 ${CLFS}/var/log/lastlog
chmod -v 600 ${CLFS}/var/log/btmp
```

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

7.17. Creating the /etc/fstab File

The `/etc/fstab` file is used by some programs to determine where file systems are to be mounted by default, which must be checked, and in which order. Create a new file systems table like this:

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

# file system  mount-point  type   options          dump  fsck
#                                                        order

/dev/[xxx]     /            [fff]  defaults         1     1
/dev/[yyy]     swap         swap   pri=1            0     0
devpts         /dev/pts     devpts gid=5,mode=620   0     0
shm            /dev/shm     tmpfs  defaults         0     0

# End /etc/fstab
EOF
```

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

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

7.18. Setting Up the Environment

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

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

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

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

7.19. Changing Ownership

Currently, the `/tools` and `/cross-tools` directories are owned by the user *clfs*, a user that exists only on the host system. Although `/tools` and `/cross-tools` can be deleted once the CLFS system has been finished, they can be retained to build additional CLFS systems. If the `/tools` and `/cross-tools` directories are kept as is, the files are owned by a user ID without a corresponding account. This is dangerous because a user account created later could get this same user ID and would own these directories and all the files therein, thus exposing those files to possible malicious manipulation.

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

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

7.20. How to View the Book

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

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

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

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

7.21. Making the Temporary System Bootable

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

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

You must now ensure that `/tools/etc/yaboot.conf` contains the correct details for the CLFS system. Consult Section 13.3, “Making the CLFS System Bootable.” for details, but note that at this point you do not need the `install, magicboot, enablecdboot` or `macosx` parameters because these are not available when you boot from Open Firmware.

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

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

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

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

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

Now set up `/tools/etc/yaboot.conf` - see Section 13.3, “Making the CLFS System Bootable.” for details of what should be in it, but note that the *install* and *magicboot* specifications should point to `/tools/lib/yaboot/` and *not* `/usr/lib/yaboot`.

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

Warning

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

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

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

```
PATH_PREFIX=/tools mkofboot
```

7.22. What to do next

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

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

Chapter 8. If You Are Going to Chroot

8.1. Introduction

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

8.2. Mounting Virtual Kernel File Systems

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

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

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

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

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

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

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

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

Now mount the remaining file systems:

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

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

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

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

8.3. Before Entering the Chroot Environment

8.3.1. Determining if steps need to be taken

Before we can enter the chroot we have to make sure that the system is in the proper state. From this point on the `${CLFS_TARGET}` environment variable will no longer exist, so it will have no bearing on the rest of the book - most packages will rely on **config.guess** provided by Section 10.36, "Automake-1.14.1". Packages that do not use autotools either do not care about the target triplet, or have their own means of determining its value.

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

If you're unsure if your host and target have the same target triplet, you can use this test to determine what the host's target triplet is and if you need to take any steps to ensure that you don't build for the wrong architecture. Extract the Section 10.36, "Automake-1.14.1" tarball and **cd** into the created directory. Then execute the following to see what the detected target triplet is by **config.guess**:

```
lib/config.guess
```

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

8.3.2. Using Setarch

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

To test if setarch does everything you need it to, execute the following command from inside the Section 10.36, "Automake-1.14.1" directory:

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

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

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

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

8.3.3. Using a Uname Hack

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

• Uname Hack (20080713) - 4 KB:
Download: *http://cross-lfs.org/files/extras/uname_hack-20080713.tar.bz2*
MD5 sum: `dd7694f28ccc6e6bfb326b1790adb5e9`

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

```
make uname_hack_fake_machine=ppc
```

The meaning of the make and install options:

uname_hack_fake_machine=ppc
: This parameter sets the value that the uts machine type will be changed to.

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

```
insmod uname_hack.ko
```

To test if the Uname Hack is working properly, execute the following command from inside the Section 10.36, “Automake-1.14.1” directory:

```
lib/config.guess
```

The output of the above command should be the same as the `${CLFS_TARGET}` environment variable. If this is not the case, you can try and get help on the CLFS Support Mailing List or the IRC Channel. See Section 1.6, “Help” for more information.

8.4. Entering the Chroot Environment

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

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

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

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

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

It is important that all the commands throughout the remainder of this chapter and the following chapters are run from within the chroot environment. If you leave this environment for any reason (rebooting for example), remember to first mount the `proc` and `devpts` file systems (discussed in the previous section) and enter chroot again before continuing with the installations.

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

8.5. Changing Ownership

Currently, the `/tools` and `/cross-tools` directories are owned by the user *clfs*, a user that exists only on the host system. Although `/tools` and `/cross-tools` can be deleted once the CLFS system has been finished, they can be retained to build additional CLFS systems. If the `/tools` and `/cross-tools` directories are kept as is, the files are owned by a user ID without a corresponding account. This is dangerous because a user account created later could get this same user ID and would own these directories and all the files therein, thus exposing those files to possible malicious manipulation.

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

```
chown -Rv 0:0 /tools
chown -Rv 0:0 /cross-tools
```

The commands use *`0:0`* instead of *`root:root`*, because **chown** is unable to resolve the name “root” until the `passwd` file has been created.

8.6. Creating Directories

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

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

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

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

8.6.1. FHS Compliance Note

The directory tree is based on the Filesystem Hierarchy Standard (FHS) (available at *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.

8.7. Creating Essential Symlinks

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

```
ln -sv /tools/bin/{bash,cat,echo,grep,pwd,stty} /bin
ln -sv /tools/bin/file /usr/bin
ln -sv /tools/lib/libgcc_s.so{,.1} /usr/lib
ln -sv /tools/lib/libstdc++.so{.6,} /usr/lib
sed -e 's/tools/usr/' /tools/lib/libstdc++.la > /usr/lib/libstdc++.la
ln -sv bash /bin/sh
```

The purpose of each link:

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

`/bin/cat`
: This pathname is hard-coded into Glibc's configure script.

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

`/bin/grep`
: This to avoid a hard-coded `/tools` reference in Libtool.

`/bin/pwd`
: Some **configure** scripts, particularly Glibc's, have this pathname hard-coded.

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

`/usr/bin/file`
: Binutils' **configure** scripts specify this command location.

`/usr/lib/libgcc_s.so{,.1}`
: Glibc needs this for the pthreads library to work.

`/usr/lib/libstdc++{,.6}`
: This is needed by several tests in Glibc's test suite, as well as for C++ support in GMP.

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

`/bin/sh`
: Many shell scripts hard-code `/bin/sh`.

`/sbin/init`
: This is where the kernel expects to find **init**.

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

```
ln -sv /proc/self/mounts /etc/mtab
```

8.8. Creating the passwd, 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
bin:x:1:1:/bin:/bin/false
daemon:x:2:6:/sbin:/bin/false
messagebus:x:27:27:D-Bus Message Daemon User:/dev/null:/bin/false
nobody:x:65534:65533:Unprivileged User:/dev/null:/bin/false
EOF
```

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

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

`adm:x:3:16:adm:/var/adm:/bin/false`
: Was used for programs that performed administrative tasks.

`lp:x:10:9:lp:/var/spool/lp:/bin/false`
: Used by programs for printing

`mail:x:30:30:mail:/var/mail:/bin/false`
: Often used by email programs

`messagebus:x:27:27:D-Bus Message Daemon User:/dev/null:/bin/false`
: User for D-Bus

`news:x:31:31:news:/var/spool/news:/bin/false`
: Often used for network news servers

`operator:x:50:0:operator:/root:/bin/bash`
: Often used to allow system operators to access the system

`postmaster:x:51:30:postmaster:/var/spool/mail:/bin/false`
: Generally used as an account that receives all the information of troubles with the mail server

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

```
cat > /etc/group << "EOF"
root:x:0:
bin:x:1:
sys:x:2:
kmem:x:3:
tty:x:5:
tape:x:4:
daemon:x:6:
floppy:x:7:
disk:x:8:
lp:x:9:
dialout:x:10:
audio:x:11:
video:x:12:
utmp:x:13:
usb:x:14:
cdrom:x:15:
adm:x:16:
messagebus:x:27:
mail:x:30:
wheel:x:39:
nogroup:x:65533:
EOF
```

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

`console:x:17:`
: This group has direct access to the console

`cdrw:x:18:`
: This group is allowed to use the CDRW drive

`messagebus:x:27:`
: User for D-Bus

`news:x:31:news`
: Used by Network News Servers

`users:x:1000:`
: The default GID used by shadow for new users

`nobody:x:65533:`
: This is used by NFS

The created groups are not part of any standard—they are groups decided on in part by the requirements of the Systemd configuration in the final system, and in part by common convention employed by a number of existing Linux distributions. The Linux Standard Base (LSB, available at *http://www.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 /var/log/{btmp,lastlog,wtmp}
chgrp -v utmp /var/log/lastlog
chmod -v 664 /var/log/lastlog
chmod -v 600 /var/log/btmp
```

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

To remove the “I have no name!” prompt, start a new shell. Since a full Glibc was installed in Constructing Cross-Compile Tools and the `/etc/passwd` and `/etc/group` files have been created, user name and group name resolution will now work.

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

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

Part V. Building the CLFS System

Chapter 9. Constructing Testsuite Tools

9.1. Introduction

This chapter builds the tools needed by some packages to run the tests that they have. I.e., **make check**. Tcl, Expect, and DejaGNU are needed for the GCC, Binutils, and Findutils test suites. Installing three packages for testing purposes may seem excessive, but it is very reassuring, if not essential, to know that the most important tools are working properly.

9.2. Tcl-8.6.1

The Tcl package contains the Tool Command Language.

9.2.1. Installation of Tcl

Increase memory size for regular expressions where required.

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

Prepare Tcl for compilation:

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

Build the package:

```
make
```

Install the package:

```
make install
```

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

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

Now make a necessary symbolic link:

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

9.2.2. Contents of Tcl

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

Short Descriptions

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

9.3. Expect-5.45

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

9.3.1. Installation of Expect

Now prepare Expect for compilation:

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

The meaning of the configure options:

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

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

Build the package:

```
make
```

Install the package:

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

The meaning of the make parameter:

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

9.3.2. Contents of Expect

Installed program: expect
Installed library: libexpect-5.43.a

Short Descriptions

expect	Communicates with other interactive programs according to a script
`libexpect-5.43.a`	Contains functions that allow Expect to be used as a Tcl extension or to be used directly from C or C++ (without Tcl)

9.4. DejaGNU-1.5.1

The DejaGNU package contains a framework for testing other programs.

9.4.1. Installation of DejaGNU

Prepare DejaGNU for compilation:

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

Build and install the package:

```
make install
```

9.4.2. Contents of DejaGNU

Installed program: runtest

Short Descriptions

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

Chapter 10. Installing Basic System Software

10.1. Introduction

In this chapter, we enter the building site and start constructing the CLFS system in earnest. The installation of this software is straightforward. Although in many cases the installation instructions could be made shorter and more generic, we have opted to provide the full instructions for every package to minimize the possibilities for mistakes. The key to learning what makes a Linux system work is to know what each package is used for and why the user (or the system) needs it. For every installed package, a summary of its contents is given, followed by concise descriptions of each program and library the package installed.

If using compiler optimizations, please review the optimization hint at *http://hints.cross-lfs.org/index.php/Optimization*. Compiler optimizations can make a program run slightly faster, but they may also cause compilation difficulties and problems when running the program. If a package refuses to compile when using optimization, try to compile it without optimization and see if that fixes the problem. Even if the package does compile when using optimization, there is the risk it may have been compiled incorrectly because of the complex interactions between the code and build tools. Also note that the `-march` and `-mtune` options may cause problems with the toolchain packages (Binutils, GCC and Glibc). The small potential gains achieved in using compiler optimizations are often outweighed by the risks. First-time builders of CLFS are encouraged to build without custom optimizations. The subsequent system will still run very fast and be stable at the same time.

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

To keep track of which package installs particular files, a package manager can be used. For a general overview of different styles of package managers, please take a look at the next page.

10.2. Package Management

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

Some reasons why no specific package manager is recommended in CLFS or CBLFS include:

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

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

10.2.1. Upgrade Issues

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

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

10.2.2. Package Management Techniques

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

10.2.2.1. It is All in My Head!

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

10.2.2.2. Install in Separate Directories

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

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

10.2.2.3. Symlink Style Package Management

This is a variation of the previous package management technique. Each package is installed similar to the previous scheme. But instead of making the symlink, each file is symlinked into the `/usr` hierarchy. This removes the need to expand the environment variables. Though the symlinks can be created by the user to automate the creation, many package managers have been written using this approach. A few of the popular ones include Stow, Epkg, Graft, and Depot.

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

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

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

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

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

10.2.2.4. Timestamp Based

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

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

10.2.2.5. LD_PRELOAD Based

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

10.2.2.6. Creating Package Archives

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

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

10.3. About Test Suites, Again

In the final-system build, you are no longer cross-compiling so it is possible to run package test suites. Running the test suite for a newly built package is a good idea because it can provide a “sanity check” indicating that everything compiled correctly. A test suite that passes its set of checks usually proves that the package is functioning as the developer intended. It does not, however, guarantee that the package is totally bug free.

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

A common issue with running the test suites for Binutils and GCC is running out of pseudo terminals (PTYs). This can result in a high number of failing tests. This may happen for several reasons, but the most likely cause (if you chrooted) is that the host system does not have the `devpts` file system set up correctly. This issue is discussed in greater detail at *http://trac.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//* to verify whether or not these failures are expected. This site is valid for all tests throughout this book.

10.4. Temporary Perl-5.20.0

The Perl package contains the Practical Extraction and Report Language.

10.4.1. Installation of Perl

> **Note**
>
> In this section, we will add Perl to the temporary system in `/tools`. This package installation should technically be part of Constructing a Temporary System, but Perl has often had problems with cross-compiling, so we will compile and install it while in the final build environment.

Change a hardcoded path from `/usr/include` to `/tools/include`:

```
sed -i 's@/usr/include@/tools/include@g' ext/Errno/Errno_pm.PL
```

Prepare Temporary Perl for compilation:

```
./configure.gnu --prefix=/tools -Dcc="gcc"
```

The meaning of the configure option:

-Dcc="gcc"
: Tells Perl to use **gcc** instead of the default **cc**.

Compile the package:

```
make
```

Although Perl comes with a test suite, it is not recommended to run it at this point, as this Perl installation is only temporary. The test suite can be run later in this chapter if desired.

Install the package:

```
make install
```

Finally, create a necessary symlink:

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

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

10.5. Linux-3.14.21 Headers

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

10.5.1. Installation of Linux Headers

> **Note**
>
> For this step you will need to unpack the kernel tarball (`linux-3.14.tar.xz`) and **cd** into its source directory before entering the commands on this page.

Apply the latest Linux sublevel patch:

```
xzcat ../patch-3.14.21.xz | patch -Np1 -i -
```

Install the kernel header files:

```
make mrproper
make headers_check
make INSTALL_HDR_PATH=/usr headers_install
find /usr/include -name .install -or -name ..install.cmd | xargs rm -fv
```

The meaning of the make commands:

`make mrproper`
: Ensures that the kernel source dir is clean.

`make headers_check`
: Sanitizes the raw kernel headers so that they can be used by userspace programs.

`make INSTALL_HDR_PATH=/usr headers_install`
: This will install the kernel headers into `/usr/include`.

`find /usr/include -name .install -or -name ..install.cmd | xargs rm -fv`
: Removes a number of unneeded debugging files that were installed.

10.5.2. Contents of Linux Headers

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

Short Descriptions

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

10.6. Man-pages-3.68

The Man-pages package contains over 2,200 man pages.

10.6.1. Installation of Man-pages

Install Man-pages by running:

```
make install
```

10.6.2. Contents of Man-pages

Installed files: various man pages

Short Descriptions

`man pages` This package contains man pages that describe the following: POSIX headers (section 0p), POSIX utilities (section 1p), POSIX functions (section 3p), user commands (section 1), system calls (section 2), libc calls (section 3), device information (section 4), file formats (section 5), games (section 6), conventions and macro packages (section 7), system administration (section 8), and kernel (section 9).

10.7. Glibc-2.19

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

10.7.1. Installation of Glibc

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

At the end of the installation, the build system will run a sanity test to make sure everything installed properly. This script performs its tests by attempting to compile test programs against certain libraries. However it does not specify the path to `ld.so`, and our toolchain is still configured to use the one in `/tools`. The following set of commands will force the script to use the complete path of the new `ld.so` that was just installed:

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

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

Apply the following **sed** so the **tzselect** script works properly:

```
sed -i 's/\\$$(pwd)/`pwd`/' timezone/Makefile
```

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

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

Prepare Glibc for compilation:

```
../glibc-2.19/configure --prefix=/usr \
    --disable-profile --enable-kernel=2.6.32 --libexecdir=/usr/lib/glibc \
    --enable-obsolete-rpc
```

The meaning of the new configure option:

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

This changes the location for hard links to the **getconf** utility from their default of `/usr/libexec` to `/usr/lib/glibc`.

Compile the package:

```
make
```

Important

Due to Glibc's critical role in a properly functioning system, the CLFS developers strongly recommend running the testsuite.

Use the following commands to run the test suite and output any test failures:

```
make -k check 2>&1 | tee glibc-check-log; 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* and *conform/run-conformtest* tests normally fail and you should see `Error 1 (ignored)` in the output. Apart from this, the Glibc test suite is always expected to pass. However, in certain circumstances, some failures are unavoidable. If a test fails because of a missing program (or missing symbolic link), or a segfault, you will see an error code greater than 127 and the details will be in the log. More commonly, tests will fail with `Error 2` - for these, the contents of the corresponding `.out` file, e.g. `posix/annexc.out` may be informative. Here is a list of the most common issues:

- The *nptl/tst-clock2*, *nptl/tst-attr3*, *tst/tst-cputimer1*, and *rt/tst-cpuclock2* tests have been known to fail. The reason is not completely understood, but indications are that minor timing issues can trigger these failures.
- The *math* tests sometimes fail. Certain optimization settings are known to be a factor here.
- If you have mounted the CLFS partition with the *`noatime`* option, the *atime* test will fail. As mentioned in Section 2.5, “Mounting the New Partition”, do not use the *`noatime`* option while building CLFS.
- When running on older and slower hardware, some tests can fail because of test timeouts being exceeded. Modifying the make check command to set a TIMEOUTFACTOR is reported to help eliminate these errors (e.g. **TIMEOUTFACTOR=16 make -k check**).
- posix/tst-getaddrinfo4 will always fail due to not having a network connection when the test is run.

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

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

Install the package, and remove unneeded files from `/usr/include/rpcsvc`:

```
make install &&
rm -v /usr/include/rpcsvc/*.x
```

Install the configuration file and runtime directory for **nscd**:

```
cp -v ../glibc-2.19/nscd/nscd.conf /etc/nscd.conf
mkdir -pv /var/cache/nscd
```

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-2.19/localedata/SUPPORTED` file) is to install only those locales that are wanted and needed. This can be achieved by using the **localedef** command. Information on this command is located in the `INSTALL` file in the Glibc source. However, there are a number of locales that are essential in order for the tests of future packages to pass, in particular, the *libstdc++* tests from GCC. The following instructions, instead of the *`install-locales`* target used above, will install the minimum set of locales necessary for the tests to run successfully:

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

Some locales installed by the **make localedata/install-locales** command above are not properly supported by some applications that are in CLFS and CBLFS. Because of the various problems that arise due to application programmers making assumptions that break in such locales, CLFS should not be used in locales that utilize multibyte character sets (including UTF-8) or right-to-left writing order. Numerous unofficial and unstable patches are required to fix these problems, and it has been decided by the CLFS developers not to support such complex locales at this time. This applies to the ja_JP and fa_IR locales as well—they have been installed only for GCC and Gettext tests to pass, and the **watch** program (part of the Procps-ng package) does not work properly in them. Various attempts to circumvent these restrictions are documented in internationalization-related hints.

10.7.3. Configuring Glibc

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

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

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

passwd: files
group: files
shadow: files

hosts: files dns
networks: files

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

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

Install timezone data:

```
tar -xf ../tzdata2014d.tar.gz

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

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

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

The meaning of the zic commands:

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

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

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

`zic ... -p ...`

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

To determine the local time zone, run the following script:

```
tzselect
```

After answering a few questions about the location, the script will output the name of the time zone (e.g., *EST5EDT* or *Canada/Eastern*). Then create the `/etc/localtime` file by running:

```
cp -v /usr/share/zoneinfo/[xxx] \
    /etc/localtime
```

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

10.7.4. Configuring The Dynamic Loader

By default, the dynamic loader (`/lib/ld.so.1`) searches through `/lib` and `/usr/lib` for dynamic libraries that are needed by programs as they are run. However, if there are libraries in directories other than `/lib` and `/usr/lib`, these need to be added to the `/etc/ld.so.conf` file in order for the dynamic loader to find them. Two directories that are commonly known to contain additional libraries are `/usr/local/lib` and `/opt/lib`, so add those directories to the dynamic loader's search path.

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

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

/usr/local/lib
/opt/lib

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

10.7.5. Contents of Glibc

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

Short Descriptions

catchsegv	Can be used to create a stack trace when a program terminates with a segmentation fault
gencat	Generates message catalogues
getconf	Displays the system configuration values for file system specific variables
getent	Gets entries from an administrative database
iconv	Performs character set conversion
iconvconfig	Creates fastloading **iconv** module configuration files
ldconfig	Configures the dynamic linker runtime bindings
ldd	Reports which shared libraries are required by each given program or shared library
lddlibc4	Assists **ldd** with object files
locale	Tells the compiler to enable or disable the use of POSIX locales for built-in operations
localedef	Compiles locale specifications
makedb	Creates a simple database from textual input
mtrace	Reads and interprets a memory trace file and displays a summary in human-readable format
nscd	A daemon that provides a cache for the most common name service requests
pcprofiledump	Dumps information generated by PC profiling
pldd	Lists dynamic shared objects used by running processes
rpcgen	Generates C code to implement the Remote Procecure Call (RPC) protocol
sln	A statically linked program that creates symbolic links
sotruss	Traces shared library procedure calls of a specified command
sprof	Reads and displays shared object profiling data

tzselect	Asks the user about the location of the system and reports the corresponding time zone description
xtrace	Traces the execution of a program by printing the currently executed function
zdump	The time zone dumper
zic	The time zone compiler
`ld.so`	The helper program for shared library executables
`libBrokenLocale`	Used by programs, such as Mozilla, to solve broken locales
`libSegFault`	The segmentation fault signal handler
`libanl`	An asynchronous name lookup library
`libc`	The main C library
`libcidn`	Used internally by Glibc for handling internationalized domain names in the `getaddrinfo()` function
`libcrypt`	The cryptography library
`libdl`	The dynamic linking interface library
`libg`	A runtime library for **g++**
`libieee`	The Institute of Electrical and Electronic Engineers (IEEE) floating point library
`libm`	The mathematical library
`libmcheck`	Contains code run at boot
`libmemusage`	Used by **memusage** (included in Glibc, but not built in a base CLFS system as it has additional dependencies) to help collect information about the memory usage of a program
`libnsl`	The network services library
`libnss`	The Name Service Switch libraries, containing functions for resolving host names, user names, group names, aliases, services, protocols, etc.
`libpcprofile`	Contains profiling functions used to track the amount of CPU time spent in specific source code lines
`libpthread`	The POSIX threads library
`libresolv`	Contains functions for creating, sending, and interpreting packets to the Internet domain name servers
`librpcsvc`	Contains functions providing miscellaneous RPC services
`librt`	Contains functions providing most of the interfaces specified by the POSIX.1b Realtime Extension
`libthread_db`	Contains functions useful for building debuggers for multi-threaded programs
`libutil`	Contains code for “standard” functions used in many different Unix utilities

10.8. Adjusting the Toolchain

Now we adjust GCC's specs so that they point to the new dynamic linker. A **perl** command accomplishes this:

```
gcc -dumpspecs | \
perl -p -e 's@/tools/lib/ld@/lib/ld@g;' \
     -e 's@\*startfile_prefix_spec:\n@$_/usr/lib/ @g;' > \
     $(dirname $(gcc --print-libgcc-file-name))/specs
```

The **perl** command above makes 2 modifications to GCC's specs: it removes “/tools” from the pathname to the dynamic linker, and adds “/usr/lib/” to the startfile_prefix_spec. It is a good idea to visually inspect the `specs` file, and compare with the output of **gcc -dumpspecs**, to verify that the intended changes were actually made.

Caution

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

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

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

```
[Requesting program interpreter: /lib/ld.so.1]
```

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

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

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

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

10.9. M4-1.4.17

The M4 package contains a macro processor.

10.9.1. Installation of M4

Prepare M4 for compilation:

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

Compile the package:

```
make
```

To test the results, issue:

```
make check
```

Install the package:

```
make install
```

10.9.2. Contents of M4

Installed program: m4

Short Descriptions

m4 copies the given files while expanding the macros that they contain. These macros are either built-in or user-defined and can take any number of arguments. Besides performing macro expansion, **m4** has built-in functions for including named files, running Unix commands, performing integer arithmetic, manipulating text, recursion, etc. The **m4** program can be used either as a front-end to a compiler or as a macro processor in its own right.

10.10. GMP-6.0.0

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

10.10.1. Installation of GMP

Note

If you are compiling this package on a different CPU than you plan to run the CLFS system on, you must replace GMP's `config.guess` and `config.sub` wrappers with the originals. This will prevent GMP from optimizing for the wrong CPU. You can make this change with the following command:

```
mv -v config{fsf,}.guess
mv -v config{fsf,}.sub
```

Prepare GMP for compilation:

```
CC="gcc -isystem /usr/include" \
CXX="g++ -isystem /usr/include" \
LDFLAGS="-Wl,-rpath-link,/usr/lib:/lib" \
  ./configure --prefix=/usr --enable-cxx \
  --docdir=/usr/share/doc/gmp-6.0.0
```

Compile the package:

```
make
```

Build the HTML documentation:

```
make html
```

Test the results:

```
make check
```

Install the package:

```
make install
```

Install the documentation:

```
make install-html
```

10.10.2. Contents of GMP

Installed libraries:	libgmp.[a,so], libgmpxx.[a,so]
Installed directory:	/usr/share/doc/gmp-6.0.0

Short Descriptions

`libgmp`	Contains the definitions for GNU multiple precision functions.
`libgmpxx`	Contains a C++ class wrapper for GMP types.

10.11. MPFR-3.1.2

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

10.11.1. Installation of MPFR

Apply a patch with upstream fixes:

```
patch -Np1 -i ../mpfr-3.1.2-fixes-4.patch
```

Prepare MPFR for compilation:

```
CC="gcc -isystem /usr/include" \
    LDFLAGS="-Wl,-rpath-link,/usr/lib:/lib" \
    ./configure --prefix=/usr --with-gmp=/usr \
    --docdir=/usr/share/doc/mpfr-3.1.2
```

Compile the package:

```
make
```

Test the results:

```
make check
```

Install the package:

```
make install
```

10.11.2. Contents of MPFR

Installed libraries: libmpfr.[a,so]
Installed directory: /usr/share/doc/mpfr-3.1.2

Short Descriptions

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

10.12. MPC-1.0.2

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

10.12.1. Installation of MPC

Prepare MPC for compilation:

```
CC="gcc -isystem /usr/include" \
LDFLAGS="-Wl,-rpath-link,/usr/lib:/lib" \
  ./configure --prefix=/usr --docdir=/usr/share/doc/mpc-1.0.2
```

Compile the package:

```
make
```

Build the HTML documentation:

```
make html
```

Test the results:

```
make check
```

Install the package:

```
make install
```

Install the HTML documentation:

```
make install-html
```

10.12.2. Contents of MPC

Installed libraries: libmpc.[a,so]
Installed directory: /usr/share/doc/mpc-1.0.2

Short Descriptions

`libmpc` The Multiple Precision Complex Library.

10.13. ISL-0.12.2

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

10.13.1. Installation of ISL

Prepare ISL for compilation:

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

Compile the package:

```
make
```

Test the results:

```
make check
```

Install the package:

```
make install
```

Finally, move a misplaced file:

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

10.13.2. Contents of ISL

Installed libraries:	libisl.[a,so]
Installed directory:	/usr/include/isl

Short Descriptions

`libisl`	The Integer Set Library.

10.14. CLooG-0.18.2

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

10.14.1. Installation of CLooG

Prepare CLooG for compilation:

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

Apply a **sed** which prevents the attempted installation of an invalid file:

```
sed -i '/cmake/d' Makefile
```

Compile the package:

```
make
```

Test the results:

```
make check
```

Install the package:

```
make install
```

10.14.2. Contents of CLooG

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

Short Descriptions

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

10.15. Zlib-1.2.8

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

10.15.1. Installation of Zlib

Prepare Zlib for compilation:

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

Compile the package:

```
make
```

To test the results, issue:

```
make check
```

Install the package:

```
make install
```

The previous command installed two `.so` files into `/usr/lib`. We will move them into `/lib` and then recreate a link in `/usr/lib`:

```
mv -v /usr/lib/libz.so.* /lib
ln -sfv ../../lib/$(readlink /usr/lib/libz.so) /usr/lib/libz.so
```

Install the documentation:

```
mkdir -pv /usr/share/doc/zlib-1.2.8
cp -rv doc/* examples /usr/share/doc/zlib-1.2.8
```

10.15.2. Contents of Zlib

Installed libraries: libz.[a,so]
Installed directory: /usr/share/doc/zlib-1.2.8

Short Descriptions

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

10.16. Flex-2.5.39

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

10.16.1. Installation of Flex

Prepare Flex for compilation:

```
./configure --prefix=/usr --docdir=/usr/share/doc/flex-2.5.39
```

Compile the package:

```
make
```

To test the results, issue:

```
make check
```

The test suite will report 3 failures for tests that use **bison**, which is not installed yet. For full test coverage, you can run Flex's test suite again after Bison is installed.

Install the package:

```
make install
```

A few programs do not know about **flex** yet and try to run its predecessor, **lex**. To support those programs, create a 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
```

10.16.2. Contents of Flex

Installed programs:	flex, flex++ (link to flex), lex
Installed libraries:	libfl.[a,so], libfl_pic.[a,so]
Installed directory:	/usr/share/doc/flex-2.5.39

Short Descriptions

flex	A tool for generating programs that recognize patterns in text; it allows for the versatility to specify the rules for pattern-finding, eradicating the need to develop a specialized program
flex++	Link to **flex** which makes it generate C++ scanner classes
lex	A script that runs **flex** in **lex** emulation mode
`libfl`	The `flex` library

`libfl_pic` The `flex` library

10.17. Bison-3.0.2

The Bison package contains a parser generator.

10.17.1. Installation of Bison

Prepare Bison for compilation:

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

Compile the package:

```
make
```

To test the results, issue:

```
make check
```

Install the package:

```
make install
```

10.17.2. Contents of Bison

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

Short Descriptions

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

10.18. Binutils-2.24

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

10.18.1. Installation of Binutils

Verify that the PTYs are working properly inside the build environment. Check that everything is set up correctly by performing a simple test:

```
expect -c "spawn ls"
```

This command should give the following output:

```
spawn ls
```

If, instead, it gives a message saying to create more ptys, then the environment is not set up for proper PTY operation. This issue needs to be resolved before running the test suites for Binutils and GCC.

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

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

Prepare Binutils for compilation:

```
CC="gcc -isystem /usr/include" \
LDFLAGS="-Wl,-rpath-link,/usr/lib:/lib" \
  ../binutils-2.24/configure --prefix=/usr \
    --enable-shared
```

Compile the package:

```
make tooldir=/usr
```

The meaning of the make parameter:

`tooldir=/usr`
: Normally, the tooldir (the directory where the executables will ultimately be located) is set to `$(exec_prefix)/$(target_alias)`. Because this is a custom system, this target-specific directory in `/usr` is not required.

Important

Due to Binutils' critical role in a properly functioning system, the CLFS developers strongly recommend running the testsuite.

Test the results:

```
make check
```

Install the package:

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

10.18.2. Contents of Binutils

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

Short Descriptions

addr2line	Translates program addresses to file names and line numbers; given an address and the name of an executable, it uses the debugging information in the executable to determine which source file and line number are associated with the address
ar	Creates, modifies, and extracts from archives
as	An assembler that assembles the output of **gcc** into object files
c++filt	Used by the linker to de-mangle C++ and Java symbols and to keep overloaded functions from clashing
elfedit	Updates the ELF header of ELF files
gprof	Displays call graph profile data
ld	A linker that combines a number of object and archive files into a single file, relocating their data and tying up symbol references
ld.bfd	Hard link to **ld**
nm	Lists the symbols occurring in a given object file
objcopy	Translates one type of object file into another
objdump	Displays information about the given object file, with options controlling the particular information to display; the information shown is useful to programmers who are working on the compilation tools
ranlib	Generates an index of the contents of an archive and stores it in the archive; the index lists all of the symbols defined by archive members that are relocatable object files
readelf	Displays information about ELF type binaries
size	Lists the section sizes and the total size for the given object files
strings	Outputs, for each given file, the sequences of printable characters that are of at least the specified length (defaulting to four); for object files, it prints, by default, only the strings from the initializing and loading sections while for other types of files, it scans the entire file
strip	Discards symbols from object files
`libbfd`	The Binary File Descriptor library
`libopcodes`	A library for dealing with opcodes—the “readable text” versions of instructions for the processor; it is used for building utilities like **objdump**.

10.19. GCC-4.8.3

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

10.19.1. Installation of GCC

The following patch contains a number of updates to the 4.8.3 branch by the GCC developers:

```
patch -Np1 -i ../gcc-4.8.3-branch_update-1.patch
```

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

```
sed -i 's@\./fixinc\.sh@-c true@' gcc/Makefile.in
```

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

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

Prepare GCC for compilation:

```
SED=sed CC="gcc -isystem /usr/include" \
CXX="g++ -isystem /usr/include" \
LDFLAGS="-Wl,-rpath-link,/usr/lib:/lib" \
  ../gcc-4.8.3/configure --prefix=/usr \
    --libexecdir=/usr/lib --enable-threads=posix \
    --enable-__cxa_atexit --enable-clocale=gnu --enable-languages=c,c++ \
    --disable-multilib --disable-libstdcxx-pch \
    --with-system-zlib --enable-checking=release --enable-libstdcxx-time
```

The meaning of the new configure option:

SED=sed
: This prevents a hard-coded path to `/tools/bin/sed` in the **fixincl** program.

Compile the package:

```
make
```

> **Important**
>
> Due to GCC's critical role in a properly functioning system, the CLFS developers strongly recommend running the testsuite.

Increase the stack size prior to running the tests:

```
ulimit -s 32768
```

Test the results, but do not stop at errors:

```
make -k check
```

The `-k` flag is used to make the test suite run through to completion and not stop at the first failure. The GCC test suite is very comprehensive and is almost guaranteed to generate a few failures. To receive a summary of the test suite results, run:

```
../gcc-4.8.3/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
```

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

```
cp -v ../gcc-4.8.3/include/libiberty.h /usr/include
```

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

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

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

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

Finally, move a misplaced file:

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

10.19.2. Contents of GCC

Installed programs:	c++, cc (link to gcc), cpp, g++, gcc, gcov
Installed libraries:	libasan.[a,so], libatomic.[a,so], libgcc.a, libgcc_eh.a, libgcc_s.so, libgcov.a, libgomp.[a,so], libiberty.a, libitm.[a,so], liblto_plugin.so, libmudflap.[a,so], libmudflapth.[a,so], libquadmath.[a,so], libssp.[a,so], libssp_nonshared.a, libstdc++.[a,so], libsupc++.a, libtsan.[a,so]
Installed directories:	/usr/include/c++, /usr/lib/gcc, /usr/share/gcc-4.8.3

Short Descriptions

cc	The C compiler
cpp	The C preprocessor; it is used by the compiler to expand the #include, #define, and similar statements in the source files
c++	The C++ compiler
g++	The C++ compiler
gcc	The C compiler
gcov	A coverage testing tool; it is used to analyze programs to determine where optimizations will have the most effect

`libasan`	The Address Sanitizer runtime library
`libatomic`	A GCC support runtime library for atomic operations not supported by hardware
`libgcc`	Contains run-time support for **gcc**
`libgcov`	Library that is linked into a program when **gcc** is instructed to enable profiling
`libgomp`	GNU implementation of the OpenMP API for multi-platform shared-memory parallel programming in C/C++ and Fortran
`libiberty`	Contains routines used by various GNU programs, including **getopt**, **obstack**, **strerror**, **strtol**, and **strtoul**
`libitm`	The GNU Transactional Memory Library, which provides transaction support for accesses to a process's memory
`liblto_plugin`	Runtime library for GCC's link-time optimization plugin
`libmudflap`	The libmudflap libraries are used by GCC for instrumenting pointer and array dereferencing operations.
`libquadmath`	The GCC Quad-Precision Math Libarary API
`libssp`	Contains routines supporting GCC's stack-smashing protection functionality
`libstdc++`	The standard C++ library
`libsupc++`	Provides supporting routines for the C++ programming language
`libtsan`	The Thread Sanitizer runtime library

10.20. Sed-4.2.2

The Sed package contains a stream editor.

10.20.1. Installation of Sed

Prepare Sed for compilation:

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

Compile the package:

```
make
```

Build the HTML documentation:

```
make html
```

To test the results, issue:

```
make check
```

Install the package:

```
make install
```

Install the HTML documentation:

```
make -C doc install-html
```

10.20.2. Contents of Sed

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

Short Descriptions

sed Filters and transforms text files in a single pass

10.21. Pkg-config-lite-0.28-1

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

10.21.1. Installation of Pkg-config-lite

Prepare Pkg-config-lite for compilation:

```
./configure --prefix=/usr --docdir=/usr/share/doc/pkg-config-0.28-1
```

Compile the package:

```
make
```

To test the results, issue:

```
make check
```

Install the package:

```
make install
```

10.21.2. Contents of Pkg-config-lite

Installed programs:	pkg-config
Installed directory:	/usr/share/doc/pkg-config-0.28-1

Short Descriptions

pkg-config The **pkg-config** program is used to retrieve information about installed libraries in the system. It is typically used to compile and link against one or more libraries.

10.22. Ncurses-5.9

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

10.22.1. Installation of Ncurses

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

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

Prepare Ncurses for compilation:

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

The meaning of the new configure option:

--with-manpage-format=normal
: This tells Ncurses not to compress its installed manpages.

--enable-pc-files
: This tells Ncurses to generate and install .pc files for **pkg-config**.

Compile the package:

```
make
```

This package has a test suite, but it can only be run after the package is installed. The tests are in the `test/` directory. See the `README` file in that directory for details.

Install the package:

```
make install
```

Move the Ncurses static libraries to the proper location:

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

Create symlinks in `/usr/lib`:

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

Many packages that use Ncurses will compile just fine against the widechar libraries, but won't know to look for them. Create linker scripts and symbolic links to allow older and non-widec compatible programs to build properly:

```
for lib in curses ncurses form panel menu ; do
        echo "INPUT(-l${lib}w)" > /usr/lib/lib${lib}.so
        ln -sfv lib${lib}w.a /usr/lib/lib${lib}.a
done
ln -sfv libncursesw.so /usr/lib/libcursesw.so
ln -sfv libncursesw.a /usr/lib/libcursesw.a
ln -sfv libncurses++w.a /usr/lib/libncurses++.a
ln -sfv ncursesw5-config /usr/bin/ncurses5-config
```

10.22.2. Contents of Ncurses

Installed programs: captoinfo (link to tic), clear, infocmp, infotocap (link to tic), ncursesw5-config, reset (link to tset), tabs, tic, toe, tput, tset

Installed libraries: libcursesw.so (link to libncursesw.so), libformw.[a,so], libmenuw.[a,so], libncurses++w.a, libncursesw.[a,so], libpanelw.[a,so]

Installed directories: /usr/share/tabset, /usr/share/terminfo

Short Descriptions

captoinfo Converts a termcap description into a terminfo description

clear Clears the screen, if possible

infocmp Compares or prints out terminfo descriptions

infotocap Converts a terminfo description into a termcap description

ncursesw5-config Provides configuration information for ncurses

reset Reinitializes a terminal to its default values

tabs Sets and clears tab stops on a terminal

tic The terminfo entry-description compiler that translates a terminfo file from source format into the binary format needed for the ncurses library routines. A terminfo file contains information on the capabilities of a certain terminal

toe Lists all available terminal types, giving the primary name and description for each

tput Makes the values of terminal-dependent capabilities available to the shell; it can also be used to reset or initialize a terminal or report its long name

tset Can be used to initialize terminals

`libcursesw` A link to `libncursesw`

`libncursesw` Contains functions to display text in many complex ways on a terminal screen; a good example of the use of these functions is the menu displayed during the kernel's **make menuconfig**

`libformw` Contains functions to implement forms

`libmenuw` Contains functions to implement menus

`libpanelw` Contains functions to implement panels

10.23. Shadow-4.2.1

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

10.23.1. Installation of Shadow

Note

If you would like to enforce the use of strong passwords, refer to *http://cblfs.cross-lfs.org/index.php/Cracklib* for installing Cracklib prior to building Shadow. After Cracklib is installed, execute this **sed** in Shadow's source directory to correct the path to the Cracklib dictionary:

```
sed -i 's@\(DICTPATH.\).*@\1/lib/cracklib/pw_dict@' etc/login.defs
```

Finally, add `--with-libcrack` to the **configure** command below.

Disable the installation of the **groups** and **nologin** programs and their man pages, as better versions of these programs are provided by Coreutils and Util-linux:

```
sed -i src/Makefile.in \
  -e 's/groups$(EXEEXT) //' -e 's/= nologin$(EXEEXT)/= /'
find man -name Makefile.in -exec sed -i \
  -e 's/man1\/groups\.1 //' -e 's/man8\/nologin\.8 //' '{}' \;
```

Prepare Shadow for compilation:

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

The meaning of the new configure option:

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

Compile the package:

```
make
```

This package does not come with a test suite.

Install the package:

```
make install
```

Instead of using the default *DES* method, use the more secure *SHA512* method of password encryption, which also allows passwords longer than 8 characters. It is also necessary to change the obsolete `/var/spool/mail` location for user mailboxes that Shadow uses by default to the `/var/mail` location used currently. Use the following sed command to make these changes to the appropriate configuration file:

```
sed -i /etc/login.defs \
    -e 's@#\(ENCRYPT_METHOD \).*@\1SHA512@' \
    -e 's@/var/spool/mail@/var/mail@'
```

Move a misplaced program to its proper location:

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

10.23.2. Configuring Shadow

This package contains utilities to add, modify, and delete users and groups; set and change their passwords; and perform other administrative tasks. For a full explanation of what *password shadowing* means, see the `doc/HOWTO` file within the unpacked source tree. If using Shadow support, keep in mind that programs which need to verify passwords (display managers, FTP programs, pop3 daemons, etc.) must be Shadow-compliant. That is, they need to be able to work with shadowed passwords.

To enable shadowed passwords, run the following command:

```
pwconv
```

To enable shadowed group passwords, run:

```
grpconv
```

To view or change the default settings for new user accounts that you create, you can edit `/etc/default/useradd`. See **man useradd** or *http://cblfs.cross-lfs.org/index.php/Configuring_for_Adding_Users* for more information.

10.23.3. Setting the root password

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

```
passwd root
```

10.23.4. Contents of Shadow

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

Short Descriptions

chage	Used to change the maximum number of days between obligatory password changes
chfn	Used to change a user's full name and other information
chgpasswd	Used to update group passwords in batch mode
chpasswd	Used to update the passwords of an entire series of user accounts
chsh	Used to change a user's default login shell
expiry	Checks and enforces the current password expiration policy
faillog	Is used to examine the log of login failures, to set a maximum number of failures before an account is blocked, or to reset the failure count
gpasswd	Is used to add and delete members and administrators to groups
groupadd	Creates a group with the given name
groupdel	Deletes the group with the given name

groupmems Allows a user to administer his/her own group membership list without the requirement of superuser privileges

groupmod Is used to modify the given group's name or GID

grpck Verifies the integrity of the group files `/etc/group` and `/etc/gshadow`

grpconv Creates or updates the shadow group file from the normal group file

grpunconv Updates `/etc/group` from `/etc/gshadow` and then deletes the latter

lastlog Reports the most recent login of all users or of a given user

login Is used by the system to let users sign on

logoutd Is a daemon used to enforce restrictions on log-on time and ports

newgrp Is used to change the current GID during a login session

newusers Is used to create or update an entire series of user accounts

passwd Is used to change the password for a user or group account

pwck Verifies the integrity of the password files `/etc/passwd` and `/etc/shadow`

pwconv Creates or updates the shadow password file from the normal password file

pwunconv Updates `/etc/passwd` from `/etc/shadow` and then deletes the latter

sg Executes a given command while the user's GID is set to that of the given group

su Runs a shell with substitute user and group IDs

useradd Creates a new user with the given name, or updates the default new-user information

userdel Deletes the given user account

usermod Is used to modify the given user's login name, User Identification (UID), shell, initial group, home directory, etc.

vigr Edits the `/etc/group` or `/etc/gshadow` files

vipw Edits the `/etc/passwd` or `/etc/shadow` files

10.24. Util-linux-2.24.2

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

10.24.1. FHS compliance notes

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

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

10.24.2. Installation of Util-linux

Note

findmnt and **lsblk** can utilize `libudev` for WWN and serial number information. If this is desired, rebuild Util-linux after Section 10.63, “Eudev-1.7”

Prepare Util-linux for compilation:

```
./configure --enable-write --docdir=/usr/share/doc/util-linux-2.24.2
```

The meaning of the configure options:

`--enable-write`
: This option allows the **write** program to be installed.

Compile the package:

```
make
```

To test the results, issue:

```
chown -Rv nobody . &&
su nobody -s /bin/bash -c "PATH=$PATH make -k check"
```

Install the package:

```
make install
```

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

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

10.24.3. Contents of Util-linux

Installed programs:	addpart, agetty, blkdiscard, blkid, blockdev, cal, cfdisk, chcpu, chrt, col, colcrt, colrm, column, ctrlaltdel, cytune, delpart, dmesg, eject, fallocate, fdformat, fdisk, findfs, findmnt, flock, fsck, fsck.cramfs, fsck.minix, fsfreeze, fstrim, getopt, hexdump, hwclock, ionice, ipcmk, ipcrm, ipcs, isosize, kill, last, lastb (link to last), ldattach, logger, look, losetup, lsblk, lscpu, lslocks, mcookie, mesg, mkfs, mkfs.bfs, mkfs.cramfs, mkfs.minix, mkswap, more, mount, mountpoint, namei, nologin, nsenter, partx, pg, pivot_root, prlimit, raw, readprofile, rename, renice, resizepart, rev, rtcwake, script, scriptreplay, setarch, setsid, setterm, sfdisk, sulogin, swaplabel, swapoff, swapon, switch_root, tailf, taskset, ul, umount, unshare, utmpdump, uuidd, uuidgen, wall, wdctl, whereis, wipefs, write
Installed libraries:	libblkid.[a,so], libmount.[a,so], libuuid.[a,so]
Installed directories:	/usr/include/blkid, /usr/include/libmount, /usr/include/uuid, /usr/share/bash-completion, /usr/share/doc/util-linux-2.24.2/getopt, /var/lib/hwclock

Short Descriptions

addpart	Informs the kernel of a new partition
agetty	Opens a tty port, prompts for a login name, and then invokes the **login** program
blkdiscard	Discards sectors on a device
blkid	A command line utility to locate and print block device attributes
blockdev	Allows users to call block device ioctls from the command line
cal	Displays a simple calendar
cfdisk	Manipulates the partition table of the given device
chcpu	Utility to configure CPUs
chrt	Manipulates real-time attributes of a process
col	Filters out reverse line feeds
colcrt	Filters **nroff** output for terminals that lack some capabilities, such as overstriking and half-lines
colrm	Filters out the given columns
column	Formats a given file into multiple columns
ctrlaltdel	Sets the function of the Ctrl+Alt+Del key combination to a hard or a soft reset
cytune	Tunes the parameters of the serial line drivers for Cyclades cards
delpart	Asks the kernel to remove a partition
dmesg	Dumps the kernel boot messages
eject	Eject removable media
fallocate	Preallocates space to a file
fdformat	Low-level formats a floppy disk
fdisk	Manipulates the partition table of the given device
findfs	Finds a file system by label or Universally Unique Identifier (UUID)
findmnt	Lists mounted filesystems or searches for a filesystem

flock	Acquires a file lock and then executes a command with the lock held
fsck	Is used to check, and optionally repair, file systems
fsck.cramfs	Performs a consistency check on the Cramfs file system on the given device
fsck.minix	Performs a consistency check on the Minix file system on the given device
fsfreeze	Suspends and resumes access to a filesystem
fstrim	Discards unused blocks on a mounted filesystem
getopt	Parses options in the given command line
hexdump	Dumps the given file in hexadecimal or in another given format
hwclock	Reads or sets the system's hardware clock, also called the Real-Time Clock (RTC) or Basic Input-Output System (BIOS) clock
ionice	Gives and sets program I/O scheduling class and priority
ipcmk	Creates various IPC resources
ipcrm	Removes the given Inter-Process Communication (IPC) resource
ipcs	Provides IPC status information
isosize	Reports the size of an iso9660 file system
kill	Send a signal to a process
last	Shows which users last logged in (and out), searching back through the `/var/log/wtmp` file; it also shows system boots, shutdowns, and run-level changes
lastb	Shows the failed login attempts, as logged in `/var/log/btmp`
ldattach	Attaches a line discipline to a serial line
logger	Enters the given message into the system log
look	Displays lines that begin with the given string
losetup	Sets up and controls loop devices
lsblk	Prints information about block devices
lscpu	Prints CPU architecture information
lslocks	Lists local system locks
mcookie	Generates magic cookies (128-bit random hexadecimal numbers) for **xauth**
mesg	Controls whether other users can send messages to the current user's terminal
mkfs	Builds a file system on a device (usually a hard disk partition)
mkfs.bfs	Creates a Santa Cruz Operations (SCO) bfs file system
mkfs.cramfs	Creates a cramfs file system
mkfs.minix	Creates a Minix file system
mkswap	Initializes the given device or file to be used as a swap area
more	A filter for paging through text one screen at a time
mount	Attaches the file system on the given device to a specified directory in the file-system tree
mountpoint	Tells you whether or not a directory is a mount point.
namei	Shows the symbolic links in the given pathnames

nologin Displays a message that an account is not available. Designed to be used as the default shell for accounts that have been disabled

nsenter Runs a program with namespaces of other processes

partx Tells the kernel about the presence and numbering of on-disk partitions

pg Displays a text file one screen full at a time

pivot_root Makes the given file system the new root file system of the current process

prlimit Gets and sets a process' resource limits

raw Binds a Linux raw character device to a block device

readprofile Reads kernel profiling information

rename Renames the given files, replacing a given string with another

renice Alters the priority of running processes

resizepart Asks the Linux kernel to resize a partition

rev Reverses the lines of a given file

rtcwake Enters a system sleep state until a specified wakeup time

script Makes a typescript of a terminal session

scriptreplay Plays back typescripts created by **script**

setarch Changes reported architecture in new program environment and sets personality flags

setsid Runs the given program in a new session

setterm Sets terminal attributes

sfdisk A disk partition table manipulator

sulogin Allows *root* to log in; it is normally invoked by **init** when the system goes into single user mode

swaplabel Prints or changes the label or UUID of a swap area

swapoff Disables devices and files for paging and swapping

swapon Enables devices and files for paging and swapping and lists the devices and files currently in use

switch_root Switches to another filesystem as the root of the mount tree

tailf Tracks the growth of a log file. Displays the last 10 lines of a log file, then continues displaying any new entries in the log file as they are created

taskset Retrieves or sets a process's CPU affinity

ul A filter for translating underscores into escape sequences indicating underlining for the terminal in use

umount Disconnects a file system from the system's file tree

unshare Runs a program with some namespaces unshared from parent

utmpdump Displays the content of the given login file in a more user-friendly format

uuidd A daemon used by the UUID library to generate time-based UUIDs in a secure and guranteed-unique fashion.

uuidgen Creates new UUIDs. Each new UUID can reasonably be considered unique among all UUIDs created, on the local system and on other systems, in the past and in the future

wall	Writes a message to all logged-in users
wdctl	Show hardware watchdog status
whereis	Reports the location of the binary, source, and man page for the given command
wipefs	Wipes a filesystem signature from a device
write	Sends a message to the given user *if* that user has not disabled receipt of such messages
`libblkid`	Contains routines for device identification and token extraction
`libmount`	Contains routines for parsing the `/etc/fstab`, `/etc/mtab`, and `/proc/self/mountinfo` files, managing `/etc/mtab`, and configuring various mount options
`libuuid`	Contains routines for generating unique identifiers for objects that may be accessible beyond the local system

10.25. Procps-ng-3.3.9

The Procps-ng package contains programs for monitoring processes.

10.25.1. Installation of Procps-ng

Prepare procps-ng for compilation:

```
./configure --prefix=/usr --exec-prefix= \
    --libdir=/usr/lib --docdir=/usr/share/doc/procps-ng-3.3.9 \
    --disable-kill
```

The meaning of the configure options:

--disable-kill
: This switch disables building the **kill** program - a better version was installed by the Util-linux package.

Compile the package:

```
make
```

> **Note**
>
> When using the boot method, two tests will fail if the hostname is not set. If you have booted the temporary system, and want to run the test suite, run the following command:
>
> ```
> hostname clfs
> ```

If running the testsuite, first disable a test which fails when scripting does not use a tty device:

```
sed -i -r 's|(pmap_initname)\\\$|\1|' testsuite/pmap.test/pmap.exp
make check
```

Install the package:

```
make install
```

Move essential files to a location that can be found if `/usr` is not mounted:

```
mv -v /usr/bin/pidof /bin
mv -v /usr/lib/libprocps.so.* /lib
ln -sfv ../../lib/$(readlink /usr/lib/libprocps.so) /usr/lib/libprocps.so
```

10.25.2. Contents of Procps-ng

Installed programs:	free, pgrep, pidof, pkill, pmap, ps, pwdx, slabtop, sysctl, tload, top, uptime, vmstat, w, watch
Installed library:	libprocps.{a,so}
Installed directories:	/usr/include/proc, /usr/share/doc/procps-ng-3.3.9

Short Descriptions

free	Reports the amount of free and used memory (both physical and swap memory) in the system

pgrep	Looks up processes based on their name and other attributes
pidof	Reports the PIDs of the given programs
pkill	Signals processes based on their name and other attributes
pmap	Reports the memory map of the given process
ps	Lists the current running processes
pwdx	Reports the current working directory of a process
slabtop	Displays detailed kernel slab cache information in real time
sysctl	Modifies kernel parameters at run time
tload	Prints a graph of the current system load average
top	Displays a list of the most CPU intensive processes; it provides an ongoing look at processor activity in real time
uptime	Reports how long the system has been running, how many users are logged on, and the system load averages
vmstat	Reports virtual memory statistics, giving information about processes, memory, paging, block Input/Output (IO), traps, and CPU activity
w	Shows which users are currently logged on, where, and since when
watch	Runs a given command repeatedly, displaying the first screen-full of its output; this allows a user to watch the output change over time
`libprocps`	Contains the functions used by most programs in this package

10.26. E2fsprogs-1.42.9

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

10.26.1. Installation of E2fsprogs

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

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

Prepare E2fsprogs for compilation:

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

The meaning of the configure options:

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

`--enable-elf-shlibs`
: This creates the shared libraries which some programs in this package use.

`--disable-*`
: This prevents E2fsprogs from building and installing the `libuuid` and `libblkid` libraries, the `uuidd` daemon, and the **fsck** wrapper, as Util-Linux installed all of them earlier.

Compile the package:

```
make
```

To test the results, issue:

```
make check
```

Install the binaries, documentation and shared libraries:

```
make install
```

Install the static libraries and headers:

```
make install-libs
```

10.26.2. Contents of E2fsprogs

Installed programs:	badblocks, chattr, compile_et, debugfs, dumpe2fs, e2freefrag, e2fsck, e2image, e2initrd_helper, e2label, e2undo, e4defrag, filefrag, fsck.ext2, fsck.ext3, fsck.ext4, fsck.ext4dev, logsave, lsattr, mk_cmds, mke2fs, mkfs.ext2, mkfs.ext3, mkfs.ext4, mkfs.ext4dev, mklost+found, resize2fs, tune2fs
Installed libraries:	libcom_err.[a,so], libe2p.[a,so], libext2fs.[a,so], libss.[a,so], libquota.a
Installed directories:	/usr/include/e2p, /usr/include/et, /usr/include/ext2fs, /usr/include/quota, /usr/include/ss, /usr/share/et, /usr/share/ss

Short Descriptions

badblocks	Searches a device (usually a disk partition) for bad blocks
chattr	Changes the attributes on a Linux file system
compile_et	An error table compiler; it converts a table of error-code names and messages into a C source file suitable for use with the `com_err` library
debugfs	A file system debugger; it can be used to examine and change the state of an `ext2` file system
dumpe2fs	Prints the super block and blocks group information for the file system present on a given device
e2freefrag	Reports free space fragmentation information
e2fsck	Is used to check, and optionally repair `ext2`, `ext3` and `ext4`file systems
e2image	Is used to save critical `ext2` file system data to a file
e2initrd_helper	Prints the FS type of a given filesystem, given either a device name or label
e2label	Displays or changes the file system label on the `ext2` file system present on a given device
e2undo	Replays an undo log for an ext2/ext3/ext4 filesystem
e4defrag	Online defragmenter for ext4 filesystems
filefrag	Reports on how badly fragmented a particular file might be
fsck.ext2	By default checks `ext2` file systems
fsck.ext3	By default checks `ext3` file systems
fsck.ext4	By default checks `ext4` file systems
fsck.ext4dev	By default checks `ext4dev` 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 `libss` subsystem library
mke2fs	Creates an `ext2`, `ext3` or `ext4` file system on the given device
mkfs.ext2	By default creates `ext2` file systems
mkfs.ext3	By default creates `ext3` file systems
mkfs.ext4	By default creates `ext4` file systems
mkfs.ext4dev	By default creates `ext4dev` file systems

mklost+found	Used to create a `lost+found` directory on an `ext2` file system; it pre-allocates disk blocks to this directory to lighten the task of **e2fsck**
resize2fs	Can be used to enlarge or shrink an `ext2` file system
tune2fs	Adjusts tunable file system parameters on an `ext2` file system
`libcom_err`	The common error display routine
`libe2p`	Used by **dumpe2fs**, **chattr**, and **lsattr**
`libext2fs`	Contains routines to enable user-level programs to manipulate an `ext2` file system
`libquota`	Provides an interface for creating and updating quota files and ext4 superblock fields
`libss`	Used by **debugfs**

10.27. Coreutils-8.22

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

10.27.1. Installation of Coreutils

A known issue with the **uname** program from this package is that the `-p` switch always returns `unknown`. The following patch fixes this behavior for all architectures:

```
patch -Np1 -i ../coreutils-8.22-uname-1.patch
```

Now prepare Coreutils for compilation:

```
FORCE_UNSAFE_CONFIGURE=1 \
  ./configure --prefix=/usr \
    --enable-no-install-program=kill,uptime \
    --enable-install-program=hostname --libexecdir=/usr/lib
```

The meaning of the configure options:

`FORCE_UNSAFE_CONFIGURE=1`
: Forces Coreutils to compile when using the root user.

Compile the package:

```
make
```

Now the test suite is ready to be run. First, run the tests that are meant to be run as user `root`:

```
make NON_ROOT_USERNAME=nobody check-root
```

The test suite will now be run as the `nobody` user. Some tests require that the user be a member of more than one group. Add a temporary group and make the user `nobody` a part of it so that the tests are not skipped:

```
echo "dummy:x:1000:nobody" >> /etc/group
```

Fix permissions of some files so the non-root user can compile and run the tests:

```
chown -Rv nobody .
```

Then run the remainder of the tests as the `nobody` user:

```
su nobody -s /bin/bash \
    -c "PATH=$PATH make RUN_EXPENSIVE_TESTS=yes -k check || true"
```

Remove the temporary group:

```
sed -i '/dummy/d' /etc/group
```

Install the package:

```
make install
```

Move programs to the locations specified by the FHS:

```
mv -v /usr/bin/{cat,chgrp,chmod,chown,cp,date} /bin
mv -v /usr/bin/{dd,df,echo,false,hostname,ln,ls,mkdir,mknod} /bin
mv -v /usr/bin/{mv,pwd,rm,rmdir,stty,true,uname} /bin
mv -v /usr/bin/chroot /usr/sbin
```

10.27.2. Contents of Coreutils

Installed programs:	[, base64, basename, cat, chcon, chgrp, chmod, chown, chroot, cksum, comm, cp, csplit, cut, date, dd, df, dir, dircolors, dirname, du, echo, env, expand, expr, factor, false, fmt, fold, groups, head, hostid, hostname, id, install, join, link, ln, logname, ls, md5sum, mkdir, mkfifo, mknod, mktemp, mv, nice, nl, nohup, nproc, numfmt, od, paste, pathchk, pinky, pr, printenv, printf, ptx, pwd, readlink, realpath, rm, rmdir, runcon, seq, sha1sum, sha224sum, sha256sum, sha384sum, sha512sum, shred, shuf, sleep, sort, split, stat, stdbuf, stty, sum, sync, tac, tail, tee, test, timeout, touch, tr, true, truncate, tsort, tty, uname, unexpand, uniq, unlink, users, vdir, wc, who, whoami, yes
Installed library:	libstdbuf.so
Installed directory:	/usr/lib/coreutils

Short Descriptions

base64	Base64 encode/decode data and print to standard output
basename	Strips any path and a given suffix from a file name
cat	Concatenates files to standard output
chcon	Changes security context for files and directories
chgrp	Changes the group ownership of files and directories
chmod	Changes the permissions of each file to the given mode; the mode can be either a symbolic representation of the changes to make or an octal number representing the new permissions
chown	Changes the user and/or group ownership of files and directories
chroot	Runs a command with the specified directory as the `/` directory
cksum	Prints the Cyclic Redundancy Check (CRC) checksum and the byte counts of each specified file
comm	Compares two sorted files, outputting in three columns the lines that are unique and the lines that are common
cp	Copies files
csplit	Splits a given file into several new files, separating them according to given patterns or line numbers and outputting the byte count of each new file
cut	Prints sections of lines, selecting the parts according to given fields or positions
date	Displays the current time in the given format, or sets the system date
dd	Copies a file using the given block size and count, while optionally performing conversions on it
df	Reports the amount of disk space available (and used) on all mounted file systems, or only on the file systems holding the selected files
dir	Lists the contents of each given directory (the same as the **ls** command)

dircolors	Outputs commands to set the LS_COLOR environment variable to change the color scheme used by **ls**
dirname	Strips the non-directory suffix from a file name
du	Reports the amount of disk space used by the current directory, by each of the given directories (including all subdirectories) or by each of the given files
echo	Displays the given strings
env	Runs a command in a modified environment
expand	Converts tabs to spaces
expr	Evaluates expressions
factor	Prints the prime factors of all specified integer numbers
false	Does nothing, unsuccessfully; it always exits with a status code indicating failure
fmt	Reformats the paragraphs in the given files
fold	Wraps the lines in the given files
groups	Reports a user's group memberships
head	Prints the first ten lines (or the given number of lines) of each given file
hostid	Reports the numeric identifier (in hexadecimal) of the host
hostname	Reports or sets the name of the host
id	Reports the effective user ID, group ID, and group memberships of the current user or specified user
install	Copies files while setting their permission modes and, if possible, their owner and group
join	Joins the lines that have identical join fields from two separate files
link	Creates a hard link with the given name to a file
ln	Makes hard links or soft (symbolic) links between files
logname	Reports the current user's login name
ls	Lists the contents of each given directory
md5sum	Reports or checks Message Digest 5 (MD5) checksums
mkdir	Creates directories with the given names
mkfifo	Creates First-In, First-Outs (FIFOs), a “named pipe” in UNIX parlance, with the given names
mknod	Creates device nodes with the given names; a device node is a character special file, a block special file, or a FIFO
mktemp	Creates temporary files in a secure manner; it is used in scripts
mv	Moves or renames files or directories
nice	Runs a program with modified scheduling priority
nl	Numbers the lines from the given files
nohup	Runs a command immune to hangups, with its output redirected to a log file
nproc	Prints the number of processing units available to the current process
numfmt	Converts numbers to or from human-readable strings
od	Dumps files in octal and other formats

paste	Merges the given files, joining sequentially corresponding lines side by side, separated by tab characters
pathchk	Checks if file names are valid or portable
pinky	Is a lightweight finger client; it reports some information about the given users
pr	Paginates and columnates files for printing
printenv	Prints the environment
printf	Prints the given arguments according to the given format, much like the C printf function
ptx	Produces a permuted index from the contents of the given files, with each keyword in its context
pwd	Reports the name of the current working directory
readlink	Reports the value of the given symbolic link
realpath	Prints the resolved path
rm	Removes files or directories
rmdir	Removes directories if they are empty
runcon	Runs a command with specified security context
seq	Prints a sequence of numbers within a given range and with a given increment
sha1sum	Prints or checks 160-bit Secure Hash Algorithm 1 (SHA1) checksums
sha224sum	Prints or checks SHA224 checksums
sha256sum	Prints or checks SHA256 checksums
sha384sum	Prints or checks SHA384 checksums
sha512sum	Prints or checks SHA512 checksums
shred	Overwrites the given files repeatedly with complex patterns, making it difficult to recover the data
shuf	Write a random permutation of the input lines to standard output or a file
sleep	Pauses for the given amount of time
sort	Sorts the lines from the given files
split	Splits the given file into pieces, by size or by number of lines
stat	Displays file or filesystem status
stdbuf	Runs a command with modified buffering operations for its standard streams
stty	Sets or reports terminal line settings
sum	Prints checksum and block counts for each given file
sync	Flushes file system buffers; it forces changed blocks to disk and updates the super block
tac	Concatenates the given files in reverse
tail	Prints the last ten lines (or the given number of lines) of each given file
tee	Reads from standard input while writing both to standard output and to the given files
test or **[**	Compares values and checks file types
timeout	Runs a command with a time limit
touch	Changes file timestamps, setting the access and modification times of the given files to the current time; files that do not exist are created with zero length

tr	Translates, squeezes, and deletes the given characters from standard input
true	Does nothing, successfully; it always exits with a status code indicating success
truncate	Shrinks or expands a file to the specified size
tsort	Performs a topological sort; it writes a completely ordered list according to the partial ordering in a given file
tty	Reports the file name of the terminal connected to standard input
uname	Reports system information
unexpand	Converts spaces to tabs
uniq	Discards all but one of successive identical lines
unlink	Removes the given file
users	Reports the names of the users currently logged on
vdir	Is the same as **ls -l**
wc	Reports the number of lines, words, and bytes for each given file, as well as a total line when more than one file is given
who	Reports who is logged on
whoami	Reports the user name associated with the current effective user ID
yes	Repeatedly outputs “y” or a given string until killed
`libstdbuf`	Library used by **stdbuf**

10.28. Iana-Etc-2.30

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

10.28.1. Installation of Iana-Etc

The following patch contains xml files which provide updates to the services and protocol files:

```
xzcat ../iana-etc-2.30-numbers_update-20140202-2.patch.xz | patch -Np1 -i -
```

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

```
make
```

This package does not come with a test suite.

Install the package:

```
make install
```

10.28.2. Contents of Iana-Etc

Installed files: /etc/protocols, /etc/services

Short Descriptions

`/etc/protocols`	Describes the various DARPA Internet protocols that are available from the TCP/IP subsystem
`/etc/services`	Provides a mapping between friendly textual names for internet services, and their underlying assigned port numbers and protocol types

10.29. Libtool-2.4.2

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

10.29.1. 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
```

10.29.2. Contents of Libtool

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

Short Descriptions

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

10.30. IPRoute2-3.14.0

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

10.30.1. Installation of IPRoute2

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

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

Compile the package:

```
make
```

This package does not come with a test suite.

Install the package:

```
make DOCDIR=/usr/share/doc/iproute2-3.14.0 install
```

10.30.2. Contents of IPRoute2

Installed programs:	bridge, ctstat (link to lnstat), genl, ifcfg, ifstat, ip, lnstat, nstat, routef, routel, rtacct, rtmon, rtpr, rtstat (link to lnstat), ss, tc
Installed directories:	/etc/iproute2, /usr/lib/tc, /usr/share/doc/iproute2-3.14.0

Short Descriptions

bridge	Configures network bridges
ctstat	Connection status utility
genl	Needs description
ifcfg	A shell script wrapper for the **ip** command
ifstat	Shows the interface statistics, including the amount of transmitted and received packets by interface
ip	The main executable. It has several different functions: **ip link `[device]`** allows users to look at the state of devices and to make changes **ip addr** allows users to look at addresses and their properties, add new addresses, and delete old ones **ip neighbor** allows users to look at neighbor bindings and their properties, add new neighbor entries, and delete old ones **ip rule** allows users to look at the routing policies and change them **ip route** allows users to look at the routing table and change routing table rules **ip tunnel** allows users to look at the IP tunnels and their properties, and change them **ip maddr** allows users to look at the multicast addresses and their properties, and change them **ip mroute** allows users to set, change, or delete the multicast routing **ip monitor** allows users to continuously monitor the state of devices, addresses and routes

lnstat Provides Linux network statistics. It is a generalized and more feature-complete replacement for the old **rtstat** program

nstat Shows network statistics

routef A component of **ip route**. This is for flushing the routing tables

routel A component of **ip route**. This is for listing the routing tables

rtacct Displays the contents of `/proc/net/rt_acct`

rtmon Route monitoring utility

rtpr Converts the output of **ip -o** back into a readable form

rtstat Route status utility

ss Similar to the **netstat** command; shows active connections

tc Traffic Controlling Executable; this is for Quality Of Service (QOS) and Class Of Service (COS) implementations

tc qdisc allows users to setup the queueing discipline

tc class allows users to setup classes based on the queuing discipline scheduling

tc estimator allows users to estimate the network flow into a network

tc filter allows users to setup the QOS/COS packet filtering

tc policy allows users to setup the QOS/COS policies

10.31. Bzip2-1.0.6

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

10.31.1. Installation of Bzip2

By default Bzip2 creates some symlinks that use absolute pathnames. The following sed will cause them to be created with relative paths instead:

```
sed -i -e 's:ln -s -f $(PREFIX)/bin/:ln -s :' Makefile
```

Make Bzip2 install its manpages in `/usr/share/man` instead of `/usr/man`:

```
sed -i 's@X)/man@X)/share/man@g' ./Makefile
```

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

```
make -f Makefile-libbz2_so
make clean
```

The *-f* flag will cause Bzip2 to be built using a different `Makefile` file, in this case the `Makefile-libbz2_so` file, which creates a dynamic `libbz2.so` library and links the Bzip2 utilities against it.

Recompile the package using a non-shared library and test it:

```
make
```

Install the programs:

```
make PREFIX=/usr install
```

Install the shared **bzip2** binary into the `/bin` directory, make some necessary symbolic links, and clean up:

```
cp -v bzip2-shared /bin/bzip2
cp -av libbz2.so* /lib
ln -sv ../../lib/libbz2.so.1.0 /usr/lib/libbz2.so
rm -v /usr/bin/{bunzip2,bzcat,bzip2}
ln -sv bzip2 /bin/bunzip2
ln -sv bzip2 /bin/bzcat
```

10.31.2. Contents of Bzip2

Installed programs:	bunzip2 (link to bzip2), bzcat (link to bzip2), bzcmp (link to bzdiff), bzdiff, bzegrep (link to bzgrep), bzfgrep (link to bzgrep), bzgrep, bzip2, bzip2recover, bzless (link to bzmore), bzmore
Installed libraries:	libbz2.a, libbz2.so (link to libbz2.so.1.0), libbz2.so.1.0 (link to libbz2.so.1.0.6), libbz2.so.1.0.6

Short Descriptions

bunzip2	Decompresses bzipped files
bzcat	Decompresses to standard output

bzcmp	Runs **cmp** on bzipped files
bzdiff	Runs **diff** on bzipped files
bzegrep	Runs **egrep** on bzipped files
bzfgrep	Runs **fgrep** on bzipped files
bzgrep	Runs **grep** on bzipped files
bzip2	Compresses files using the Burrows-Wheeler block sorting text compression algorithm with Huffman coding; the compression rate is better than that achieved by more conventional compressors using “Lempel-Ziv” algorithms, like **gzip**
bzip2recover	Tries to recover data from damaged bzipped files
bzless	Runs **less** on bzipped files
bzmore	Runs **more** on bzipped files
`libbz2*`	The library implementing lossless, block-sorting data compression, using the Burrows-Wheeler algorithm

10.32. GDBM-1.11

The GDBM package contains the GNU Database Manager. This is a disk file format database which stores key/data-pairs in single files. The actual data of any record being stored is indexed by a unique key, which can be retrieved in less time than if it was stored in a text file.

10.32.1. Installation of GDBM

Prepare GDBM for compilation:

```
./configure --prefix=/usr --enable-libgdbm-compat
```

The meaning of the configure option:

`--enable-libgdbm-compat`
: This switch enables the libgdbm compatibility library to be built, as some packages outside of CLFS may require the older DBM routines it provides.

Compile the package:

```
make
```

To test the results, issue:

```
make check
```

Install the package:

```
make install
```

10.32.2. Contents of GDBM

Installed programs: gdbm_dump, gdbm_load, gdbmtool
Installed libraries: libgdbm.{a,so}, libgdbm_compat.{a,so}

Short Descriptions

gdbm_dump	Dumps a GDBM database to a file.
gdbm_load	Recreates a GDBM database from a dump file.
gdbmtool	Tests and modifies a GDBM database
`libgdbm`	Contains functions to manipulate a hashed database
`libgdbm_compat`	Compatibility library containing older DBM functions

10.33. Perl-5.20.0

The Perl package contains the Practical Extraction and Report Language.

10.33.1. Installation of Perl

By default, Perl's Compress::Raw::Zlib and Compress::Raw::Bzip2 modules build and link against internal copies of Zlib and Bzip2. The following command will make Perl use the system-installed copies of these libraries:

```
export BUILD_ZLIB=False
export BUILD_BZIP2=0
```

Note

If you are following the boot method you will need to enable the loopback device:

```
ip link set lo up
```

Before starting to configure, create a basic `/etc/hosts` file which will be referenced by one of Perl's configuration files as well as used by the test suite:

```
echo "127.0.0.1 localhost $(hostname)" > /etc/hosts
```

To have full control over the way Perl is set up, you can run the interactive **Configure** script and hand-pick the way this package is built. If you prefer instead to use the defaults that Perl auto-detects, prepare Perl for compilation with:

```
./configure.gnu --prefix=/usr \
   -Dvendorprefix=/usr \
   -Dman1dir=/usr/share/man/man1 \
   -Dman3dir=/usr/share/man/man3 \
   -Dpager="/bin/less -isR" \
   -Dusethreads -Duseshrplib
```

The meaning of the configure option:

-Dman1dir=/usr/share/man/man1 -Dman3dir=/usr/share/man/man3
: Since Groff is not installed yet, **configure.gnu** thinks that we do not want man pages for Perl. Issuing these parameters overrides this decision.

-Dpager="/bin/less -isR"
: Less has not yet been installed, so by default **perldoc** will invoke the **more** program for viewing documentation. This option ensures that it will use **less** instead.

-Dusethreads
: This tells Perl to use threads.

-Duseshrplib
: This tells Perl to build a shared libperl.

Compile the package:

```
make
```

To test the results, issue:

```
make test
```

Install the package and remove the variables set previously:

```
make install
unset BUILD_ZLIB BUILD_BZIP2
```

10.33.2. Contents of Perl

Installed programs:	a2p, c2ph, config_data, corelist, cpan, cpan2dist, cpanp, cpanp-run-perl, enc2xs, find2perl, h2ph, h2xs, instmodsh, json_pp, libnetcfg, perl, perl5.20.0 (link to perl), perlbug, perldoc, perlivp, perlthanks (link to perlbug), piconv, pl2pm, pod2html, pod2latex, pod2man, pod2text, pod2usage, podchecker, podselect, prove, psed (link to s2p), pstruct (link to c2ph), ptar, ptardiff, ptargrep, s2p, shasum, splain, xsubpp, zipdetails
Installed libraries:	Several hundred which cannot all be listed here
Installed directory:	/usr/lib/perl5

Short Descriptions

a2p	Translates awk to Perl
c2ph	Dumps C structures as generated from **cc -g -S**
config_data	Queries or changes configuration of Perl modules
corelist	A commandline frontend to Module::CoreList
cpan	Shell script that provides a command interface to CPAN.pm
cpan2dist	The CPANPLUS distribution creator
cpanp	The CPANPLUS launcher
cpanp-run-perl	Perl script that (description needed)
enc2xs	Builds a Perl extension for the Encode module from either Unicode Character Mappings or Tcl Encoding Files
find2perl	Translates **find** commands to Perl
h2ph	Converts `.h` C header files to `.ph` Perl header files
h2xs	Converts `.h` C header files to Perl extensions
instmodsh	A shell script for examining installed Perl modules, and can even create a tarball from an installed module
json_pp	Converts data between certain input and output formats
libnetcfg	Can be used to configure the `libnet`
perl	Combines some of the best features of C, **sed**, **awk** and **sh** into a single swiss-army-knife language
perl5.20.0	A hard link to **perl**
perlbug	Used to generate bug reports about Perl, or the modules that come with it, and mail them

perldoc	Displays a piece of documentation in pod format that is embedded in the Perl installation tree or in a Perl script
perlivp	The Perl Installation Verification Procedure; it can be used to verify that Perl and its libraries have been installed correctly
perlthanks	Used to generate thank you messages to mail to the Perl developers
piconv	A Perl version of the character encoding converter **iconv**
pl2pm	A rough tool for converting Perl4 `.pl` files to Perl5 `.pm` modules
pod2html	Converts files from pod format to HTML format
pod2latex	Converts files from pod format to LaTeX format
pod2man	Converts pod data to formatted *roff input
pod2text	Converts pod data to formatted ASCII text
pod2usage	Prints usage messages from embedded pod docs in files
podchecker	Checks the syntax of pod format documentation files
podselect	Displays selected sections of pod documentation
prove	A command-line tool for running tests against Test::Harness
psed	A Perl version of the stream editor **sed**
pstruct	Dumps C structures as generated from **cc -g -S** stabs
ptar	A **tar**-like program written in Perl
ptardiff	A Perl program that compares an extracted archive with an unextracted one
ptargrep	A Perl program that applies pattern matching to the contents of files in a tar archive
s2p	Translates **sed** to Perl
shasum	Prints or checks SHA checksums
splain	Is used to force verbose warning diagnostics in Perl
xsubpp	Converts Perl XS code into C code
zipdetails	Displays details about the internal structure of a Zip file

10.34. Readline-6.3

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

10.34.1. Installation of Readline

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

```
patch -Np1 -i ../readline-6.3-branch_update-4.patch
```

Prepare Readline for compilation:

```
./configure --prefix=/usr --libdir=/lib \
    --docdir=/usr/share/doc/readline-6.3
```

Compile the package:

```
make SHLIB_LIBS=-lncurses
```

This package does not come with a test suite.

Install the package:

```
make SHLIB_LIBS=-lncurses htmldir=/usr/share/doc/readline-6.3 install
```

Now move the static libraries to a more appropriate location:

```
mv -v /lib/lib{readline,history}.a /usr/lib
```

Next, relink the dynamic libraries into `/usr/lib` and remove the `.so` files in `/lib`.

```
ln -svf ../../lib/$(readlink /lib/libreadline.so) /usr/lib/libreadline.so
ln -svf ../../lib/$(readlink /lib/libhistory.so) /usr/lib/libhistory.so
rm -v /lib/lib{readline,history}.so
```

10.34.2. Contents of Readline

Installed libraries: libhistory.[a,so], libreadline.[a,so]
Installed directories: /usr/include/readline, /usr/share/doc/readline-6.3, /usr/share/readline

Short Descriptions

`libhistory` Provides a consistent user interface for recalling lines of history

`libreadline` Aids in the consistency of user interface across discrete programs that need to provide a command line interface

10.35. Autoconf-2.69

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

10.35.1. Installation of Autoconf

Prepare Autoconf for compilation:

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

Compile the package:

```
make
```

To test the results, issue:

```
make check VERBOSE=yes
```

17 tests are skipped that use Automake and different GCC languages. For full test coverage, Autoconf can be re-tested after Automake has been installed.

Install the package:

```
make install
```

10.35.2. Contents of Autoconf

Installed programs:	autoconf, autoheader, autom4te, autoreconf, autoscan, autoupdate, ifnames
Installed directory:	/usr/share/autoconf

Short Descriptions

autoconf	Produces shell scripts that automatically configure software source code packages to adapt to many kinds of Unix-like systems. The configuration scripts it produces are independent—running them does not require the **autoconf** program.
autoheader	A tool for creating template files of C *#define* statements for configure to use
autom4te	A wrapper for the M4 macro processor
autoreconf	Automatically runs **autoconf**, **autoheader**, **aclocal**, **automake**, **gettextize**, and **libtoolize** in the correct order to save time when changes are made to **autoconf** and **automake** template files
autoscan	Helps to create a `configure.in` file for a software package; it examines the source files in a directory tree, searching them for common portability issues, and creates a `configure.scan` file that serves as as a preliminary `configure.in` file for the package
autoupdate	Modifies a `configure.in` file that still calls **autoconf** macros by their old names to use the current macro names
ifnames	Helps when writing `configure.in` files for a software package; it prints the identifiers that the package uses in C preprocessor conditionals. If a package has already been set up to have some portability, this program can help determine what **configure** needs to check for. It can also fill in gaps in a `configure.in` file generated by **autoscan**

10.36. Automake-1.14.1

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

10.36.1. Installation of Automake

Prepare Automake for compilation:

```
./configure --prefix=/usr --docdir=/usr/share/doc/automake-1.14.1
```

Compile the package:

```
make
```

To test the results, issue:

```
make check
```

Install the package:

```
make install
```

10.36.2. Contents of Automake

Installed programs: aclocal, aclocal-1.14, automake, automake-1.14, compile, config.guess, config.sub, depcomp, install-sh, mdate-sh, missing, mkinstalldirs, py-compile, symlink-tree, ylwrap

Installed directories: /usr/share/aclocal-1.14, /usr/share/automake-1.14, /usr/share/doc/automake

Short Descriptions

aclocal	Generates `aclocal.m4` files based on the contents of `configure.in` files
aclocal-1.14	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-1.14	A hard link to **automake**
compile	A wrapper for compilers
config.guess	A script that attempts to guess the canonical triplet for the given build, host, or target architecture
config.sub	A configuration validation subroutine script
depcomp	A script for compiling a program so that dependency information is generated in addition to the desired output
install-sh	A script that installs a program, script, or data file
mdate-sh	A script that prints the modification time of a file or directory
missing	A script acting as a common stub for missing GNU programs during an installation
mkinstalldirs	A script that creates a directory tree

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

10.37. Bash-4.3

The Bash package contains the Bourne-Again SHell.

10.37.1. Installation of Bash

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

```
patch -Np1 -i ../bash-4.3-branch_update-5.patch
```

Prepare Bash for compilation:

```
./configure --prefix=/usr --bindir=/bin \
    --without-bash-malloc --with-installed-readline \
    --docdir=/usr/share/doc/bash-4.3
```

The meaning of the new configure option:

--with-installed-readline
: This option tells Bash to use the `readline` library that is already installed on the system rather than using its own readline version.

Compile the package:

```
make
```

To test the results, issue:

```
make tests
```

Install the package:

```
make install
```

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.

10.37.2. Contents of Bash

Installed programs:	bash, bashbug, sh (link to bash)
Installed directory:	/usr/share/doc/bash-4.3

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

10.38. Bc-1.06.95

The Bc package contains an arbitrary precision numeric processing language.

10.38.1. Installation of Bc

Prepare Bc for compilation:

```
./configure --prefix=/usr --with-readline \
    --mandir=/usr/share/man --infodir=/usr/share/info
```

Compile the package:

```
make
```

To test the results, issue:

```
echo "quit" | ./bc/bc -l Test/checklib.b
```

Install the package:

```
make install
```

10.38.2. Contents of Bc

Installed programs: bc, dc

Short Descriptions

bc is a command line calculator

dc is a reverse-polish command line calculator

10.39. Diffutils-3.3

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

10.39.1. Installation of Diffutils

Fix a bug that prevents locale files from being installed:

```
sed -i 's:= @mkdir_p@:= /bin/mkdir -p:' po/Makefile.in.in
```

Prepare Diffutils for compilation:

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

Diffutils wants **ed** as the default editor for **sdiff**. The following **sed** will change the default to **vi**:

```
sed -i 's@\(^#define DEFAULT_EDITOR_PROGRAM \).*@\1"vi"@' lib/config.h
```

Compile the package:

```
make
```

To test the results, issue:

```
make check
```

Install the package:

```
make install
```

10.39.2. Contents of Diffutils

Installed programs: cmp, diff, diff3, sdiff

Short Descriptions

cmp	Compares two files and reports whether or in which bytes they differ
diff	Compares two files or directories and reports which lines in the files differ
diff3	Compares three files line by line
sdiff	Merges two files and interactively outputs the results

10.40. File-5.19

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

10.40.1. Installation of File

Prepare File for compilation:

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

Compile the package:

```
make
```

To test the results, issue:

```
make check
```

Install the package:

```
make install
```

10.40.2. 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

10.41. Gawk-4.1.1

The Gawk package contains programs for manipulating text files.

10.41.1. Installation of Gawk

Prepare Gawk for compilation:

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

Compile the package:

```
make
```

To test the results, issue:

```
make check
```

Install the package:

```
make install
```

Install the documentation:

```
mkdir -v /usr/share/doc/gawk-4.1.1
cp -v doc/{awkforai.txt,*.{eps,pdf,jpg}} /usr/share/doc/gawk-4.1.1
```

10.41.2. Contents of Gawk

Installed programs: awk (link to gawk), gawk, gawk-4.1.1, grcat, igawk, pgawk, pgawk-4.1.1, pwcat
Installed directories: /usr/lib/awk, /usr/lib/gawk, /usr/share/awk, /usr/share/doc/gawk-4.1.1

Short Descriptions

awk	A link to **gawk**
gawk	A program for manipulating text files; it is the GNU implementation of **awk**
gawk-4.1.1	A hard link to **gawk**
grcat	Dumps the group database `/etc/group`
igawk	Gives **gawk** the ability to include files
pgawk	The profiling version of **gawk**
pgawk-4.1.1	Hard link to **pgawk**
pwcat	Dumps the password database `/etc/passwd`

10.42. Findutils-4.4.2

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

10.42.1. Installation of Findutils

Prepare Findutils for compilation:

```
./configure --prefix=/usr --libexecdir=/usr/lib/locate \
    --localstatedir=/var/lib/locate
```

The meaning of the configure options:

--localstatedir
: This option changes the location of the **locate** database to be in `/var/lib/locate`, which is FHS-compliant.

Compile the package:

```
make
```

To test the results, issue:

```
make check
```

Install the package:

```
make install
```

10.42.2. Contents of Findutils

Installed programs: bigram, code, find, frcode, locate, oldfind, updatedb, xargs
Installed directory: /usr/lib/locate

Short Descriptions

bigram Was formerly used to produce **locate** databases

code Was formerly used to produce **locate** databases; it is the ancestor of **frcode**.

find Searches given directory trees for files matching the specified criteria

frcode Is called by **updatedb** to compress the list of file names; it uses front-compression, reducing the database size by a factor of four to five.

locate Searches through a database of file names and reports the names that contain a given string or match a given pattern

oldfind Older version of find, using a different algorithm

updatedb Updates the **locate** database; it scans the entire file system (including other file systems that are currently mounted, unless told not to) and puts every file name it finds into the database

xargs Can be used to apply a given command to a list of files

10.43. Gettext-0.19.1

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

10.43.1. Installation of Gettext

Prepare Gettext for compilation:

```
./configure --prefix=/usr --docdir=/usr/share/doc/gettext-0.19.1
```

Compile the package:

```
make
```

To test the results, issue:

```
make check
```

Install the package:

```
make install
```

10.43.2. Contents of Gettext

Installed programs:	autopoint, config.charset, config.rpath, envsubst, gettext, gettext.sh, gettextize, hostname, msgattrib, msgcat, msgcmp, msgcomm, msgconv, msgen, msgexec, msgfilter, msgfmt, msggrep, msginit, msgmerge, msgunfmt, msguniq, ngettext, recode-sr-latin, xgettext
Installed libraries:	libasprintf.[a,so], libgettextlib.so, libgettextpo.[a,so], libgettextsrc.so, preloadable_libintl.so
Installed directories:	/usr/lib/gettext, /usr/share/doc/gettext-0.19.1, /usr/share/gettext

Short Descriptions

autopoint	Copies standard Gettext infrastructure files into a source package
config.charset	Outputs a system-dependent table of character encoding aliases
config.rpath	Outputs a system-dependent set of variables, describing how to set the runtime search path of shared libraries in an executable
envsubst	Substitutes environment variables in shell format strings
gettext	Translates a natural language message into the user's language by looking up the translation in a message catalog
gettext.sh	Primarily serves as a shell function library for gettext
gettextize	Copies all standard Gettext files into the given top-level directory of a package to begin internationalizing it
hostname	Displays a network hostname in various forms
msgattrib	Filters the messages of a translation catalog according to their attributes and manipulates the attributes

msgcat	Concatenates and merges the given `.po` files
msgcmp	Compares two `.po` files to check that both contain the same set of msgid strings
msgcomm	Finds the messages that are common to to the given `.po` files
msgconv	Converts a translation catalog to a different character encoding
msgen	Creates an English translation catalog
msgexec	Applies a command to all translations of a translation catalog
msgfilter	Applies a filter to all translations of a translation catalog
msgfmt	Generates a binary message catalog from a translation catalog
msggrep	Extracts all messages of a translation catalog that match a given pattern or belong to some given source files
msginit	Creates a new `.po` file, initializing the meta information with values from the user's environment
msgmerge	Combines two raw translations into a single file
msgunfmt	Decompiles a binary message catalog into raw translation text
msguniq	Unifies duplicate translations in a translation catalog
ngettext	Displays native language translations of a textual message whose grammatical form depends on a number
recode-sr-latin	Recode Serbian text from Cyrillic to Latin script.
xgettext	Extracts the translatable message lines from the given source files to make the first translation template
`libasprintf`	defines the *autosprintf* class, which makes C formatted output routines usable in C++ programs, for use with the <*string*> strings and the <*iostream*> streams
`libgettextlib`	a private library containing common routines used by the various Gettext programs; these are not intended for general use
`libgettextpo`	Used to write specialized programs that process `.po` files; this library is used when the standard applications shipped with Gettext (such as **msgcomm**, **msgcmp**, **msgattrib**, and **msgen**) will not suffice
`libgettextsrc`	A private library containing common routines used by the various Gettext programs; these are not intended for general use
`preloadable_libintl.so`	A library, intended to be used by LD_PRELOAD, that assists `libintl` in logging untranslated messages.

10.44. Grep-2.19

The Grep package contains programs for searching through files.

10.44.1. 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
```

10.44.2. Contents of Grep

Installed programs: egrep, fgrep, grep

Short Descriptions

egrep Prints lines matching an extended regular expression

fgrep Prints lines matching a list of fixed strings

grep Prints lines matching a basic regular expression

10.45. Groff-1.22.2

The Groff package contains programs for processing and formatting text.

10.45.1. Installation of Groff

Groff expects the environment variable `PAGE` to contain the default paper size. For users in the United States, *`PAGE=letter`* is appropriate. Elsewhere, *`PAGE=A4`* may be more suitable.

Prepare Groff for compilation:

```
PAGE=[paper_size] ./configure --prefix=/usr
```

Compile the package:

```
make
```

This package does not come with a test suite.

Install the package:

```
make install
```

10.45.2. Contents of Groff

Installed programs: addftinfo, afmtodit, chem, eqn, eqn2graph, gdiffmk, grap2graph, grn, grodvi, groff, groffer, grog, grolbp, grolj4, grops, grotty, hpftodit, indxbib, lkbib, lookbib, mmroff, neqn, nroff, pdfroff, pfbtops, pic, pic2graph, post-grohtml, pre-grohtml, preconv, refer, roff2dvi, roff2html, roff2pdf, roff2ps, roff2text, roff2x, soelim, tbl, tfmtodit, troff

Installed directories: /usr/lib/groff, /usr/share/doc/groff-1.22.2, /usr/share/groff

Short Descriptions

addftinfo Reads a troff font file and adds some additional font-metric information that is used by the **groff** system

afmtodit Creates a font file for use with **groff** and **grops**

chem Groff preprocessor for producing chemical structure diagrams

eqn Compiles descriptions of equations embedded within troff input files into commands that are understood by **troff**

eqn2graph Converts a troff EQN (equation) into a cropped image

gdiffmk Marks differences between groff/nroff/troff files

grap2graph Converts a grap diagram into a cropped bitmap image

grn A **groff** preprocessor for gremlin files

grodvi A driver for **groff** that produces TeX dvi format

groff A front-end to the groff document formatting system; normally, it runs the **troff** program and a post-processor appropriate for the selected device

groffer Displays groff files and man pages on X and tty terminals

grog	Reads files and guesses which of the **groff** options `-e`, `-man`, `-me`, `-mm`, `-ms`, `-p`, `-s`, and `-t` are required for printing files, and reports the **groff** command including those options
grolbp	Is a **groff** driver for Canon CAPSL printers (LBP-4 and LBP-8 series laser printers)
grolj4	Is a driver for **groff** that produces output in PCL5 format suitable for an HP LaserJet 4 printer
grops	Translates the output of GNU **troff** to PostScript
grotty	Translates the output of GNU **troff** into a form suitable for typewriter-like devices
hpftodit	Creates a font file for use with **groff -Tlj4** from an HP-tagged font metric file
indxbib	Creates an inverted index for the bibliographic databases with a specified file for use with **refer**, **lookbib**, and **lkbib**
lkbib	Searches bibliographic databases for references that contain specified keys and reports any references found
lookbib	Prints a prompt on the standard error (unless the standard input is not a terminal), reads a line containing a set of keywords from the standard input, searches the bibliographic databases in a specified file for references containing those keywords, prints any references found on the standard output, and repeats this process until the end of input
mmroff	A simple preprocessor for **groff**
neqn	Formats equations for American Standard Code for Information Interchange (ASCII) output
nroff	A script that emulates the **nroff** command using **groff**
pdfroff	Creates pdf documents using groff
pfbtops	Translates a PostScript font in `.pfb` format to ASCII
pic	Compiles descriptions of pictures embedded within troff or TeX input files into commands understood by TeX or **troff**
pic2graph	Converts a PIC diagram into a cropped image
post-grohtml	Translates the output of GNU **troff** to HTML
pre-grohtml	Translates the output of GNU **troff** to HTML
preconv	Converts encoding of input files to something GNU **troff** understands
refer	Copies the contents of a file to the standard output, except that lines between *.[* and *.]* are interpreted as citations, and lines between *.R1* and *.R2* are interpreted as commands for how citations are to be processed
roff2dvi	Transforms roff files into other formats
roff2html	Transforms roff files into other formats
roff2pdf	Transforms roff files into other formats
roff2ps	Transforms roff files into other formats
roff2text	Transforms roff files into other formats
roff2x	Transforms roff files into other formats
soelim	Reads files and replaces lines of the form *.so file* by the contents of the mentioned *file*
tbl	Compiles descriptions of tables embedded within troff input files into commands that are understood by **troff**

tfmtodit Creates a font file for use with **groff -Tdvi**

troff Is highly compatible with Unix **troff**; it should usually be invoked using the **groff** command, which will also run preprocessors and post-processors in the appropriate order and with the appropriate options

10.46. Less-462

The Less package contains a text file viewer.

10.46.1. Installation of Less

Prepare Less for compilation:

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

Compile the package:

```
make
```

This package does not come with a test suite.

Install the package:

```
make install
```

Move **less** to `/bin`:

```
mv -v /usr/bin/less /bin
```

10.46.2. Contents of Less

Installed programs: less, lessecho, lesskey

Short Descriptions

less A file viewer or pager; it displays the contents of the given file, letting the user scroll, find strings, and jump to marks

lessecho Needed to expand meta-characters, such as * and ?, in filenames on Unix systems

lesskey Used to specify the key bindings for **less**

10.47. Gzip-1.6

The Gzip package contains programs for compressing and decompressing files.

10.47.1. Installation of Gzip

Prepare Gzip for compilation:

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

Compile the package:

```
make
```

To test the results, issue:

```
make check
```

Install the package:

```
make install
```

Now we will move some of the utilities to `/usr/bin` to meet FHS compliance:

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

10.47.2. Contents of Gzip

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

Short Descriptions

gunzip	Decompresses gzipped files
gzexe	Creates self-decompressing executable files
gzip	Compresses the given files using Lempel-Ziv (LZ77) coding
uncompress	Decompresses compressed files
zcat	Decompresses the given gzipped files to standard output
zcmp	Runs **cmp** on gzipped files
zdiff	Runs **diff** on gzipped files
zegrep	Runs **egrep** on gzipped files
zfgrep	Runs **fgrep** on gzipped files
zforce	Forces a `.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`

10.48. IPutils-s20121221

The IPutils package contains programs for basic networking.

10.48.1. Installation of IPutils

IPutils has various issues addressed by the following patch:

```
patch -Np1 -i ../iputils-s20121221-fixes-2.patch
```

Compile the package:

```
make USE_CAP=no \
    IPV4_TARGETS="tracepath ping clockdiff rdisc" \
    IPV6_TARGETS="tracepath6 traceroute6"
```

This package does not come with a test suite.

Install the package:

```
install -v -m755 ping /bin
install -v -m755 clockdiff /usr/bin
install -v -m755 rdisc /usr/bin
install -v -m755 tracepath /usr/bin
install -v -m755 trace{path,route}6 /usr/bin
install -v -m644 doc/*.8 /usr/share/man/man8
```

10.48.2. Contents of iputils

Installed programs: clockdiff, ping, rdisc, tracepath, tracepath6, traceroute6

Short Descriptions

clockdiff	Measures the clock difference between hosts
ping	Sends echo-request packets and reports how long the replies take. This is the IPV4 version
rdisc	Network router discovery daemon
tracepath	Traces the path to a network host discovering MTU along the path. This is the IPV4 version.
tracepath6	Traces the path to a network host discovering MTU along the path. This is the IPV6 version.
traceroute6	Traces the path to a network host on an IPV6 network

10.49. Kbd-2.0.1

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

10.49.1. Installation of Kbd

Prepare Kbd for compilation:

```
PKG_CONFIG_PATH="/tools/lib/pkgconfig" \
    ./configure --prefix=/usr --disable-vlock --enable-optional-progs
```

The meaning of the new configure options:

`PKG_CONFIG_PATH`
: Use pkg-config to obtain the location of the test library metadata built in Section 6.14, “Check-0.9.13”.

`--disable-vlock`
: Prevents Kbd from trying to build the **vlock** program, which requires Linux-PAM.

`--enable-optional-progs`
: Installs several additional programs.

Compile the package:

```
make
```

To test the results, issue:

```
make check
```

Install the package:

```
make install
```

Some of the programs from Kbd are used by the CLFS Bootscripts to initialize the system, so those binaries need to be on the root partition:

```
mv -v /usr/bin/{dumpkeys,kbd_mode,loadkeys,setfont} /bin
```

Install the documentation:

```
mkdir -v /usr/share/doc/kbd-2.0.1
cp -R -v docs/doc/* /usr/share/doc/kbd-2.0.1
```

10.49.2. Contents of Kbd

Installed programs:	chvt, deallocvt, dumpkeys, fgconsole, getkeycodes, kbdinfo, kbd_mode, kbdrate, loadkeys, loadunimap, mapscrn, openvt, psfaddtable (link to psfxtable), psfgettable (link to psfxtable), psfstriptable (link to psfxtable), psfxtable, resizecons, setfont, setkeycodes, setleds, setmetamode, setvtrgb, showconsolefont, showkey, unicode_start, unicode_stop
Installed directories:	/usr/share/consolefonts, /usr/share/consoletrans, /usr/share/doc/kbd-2.0.1, /usr/share/keymaps, /usr/share/unimaps

Short Descriptions

chvt	Changes the foreground virtual terminal
deallocvt	Deallocates unused virtual terminals
dumpkeys	Dumps the keyboard translation tables
fgconsole	Prints the number of the active virtual terminal
getkeycodes	Prints the kernel scancode-to-keycode mapping table
kbdinfo	Obtains information about the console
kbd_mode	Reports or sets the keyboard mode
kbdrate	Sets the keyboard repeat and delay rates
loadkeys	Loads the keyboard translation tables
loadunimap	Loads the kernel unicode-to-font mapping table
mapscrn	An obsolete program that used to load a user-defined output character mapping table into the console driver; this is now done by **setfont**
openvt	Starts a program on a new virtual terminal (VT)
psfaddtable	Adds a Unicode character table to a console font
psfgettable	Extracts the embedded Unicode character table from a console font
psfstriptable	Removes the embedded Unicode character table from a console font
psfxtable	Handle Unicode character tables for console fonts
resizecons	Changes the kernel idea of the console size
setfont	Changes the Enhanced Graphic Adapter (EGA) and Video Graphics Array (VGA) fonts on the console
setkeycodes	Loads kernel scancode-to-keycode mapping table entries; this is useful if there are unusual keys on the keyboard
setleds	Sets the keyboard flags and Light Emitting Diodes (LEDs)
setmetamode	Defines the keyboard meta-key handling
setvtrgb	Sets the virtal terminal RGB colors
showconsolefont	Shows the current EGA/VGA console screen font
showkey	Reports the scancodes, keycodes, and ASCII codes of the keys pressed on the keyboard
unicode_start	Puts the keyboard and console in UNICODE mode. Never use it on CLFS, because applications are not configured to support UNICODE.
unicode_stop	Reverts keyboard and console from UNICODE mode

10.50. Libpipeline-1.3.0

The Libpipeline package contains a library for manipulating pipelines of subprocesses in a flexible and convenient way.

10.50.1. Installation of Libpipeline

Prepare Libpipeline for compilation:

```
PKG_CONFIG_PATH=/tools/lib/pkgconfig ./configure --prefix=/usr
```

Compile the package:

```
make
```

To test the results, issue:

```
make check
```

Install the package:

```
make install
```

10.50.2. Contents of Libpipeline

Installed libraries: libpipeline.so

Short Descriptions

`libpipeline`	This library is used to safely construct pipeline between subprocesses

10.51. Man-DB-2.6.7.1

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

10.51.1. Installation of Man-DB

Prepare Man-DB for compilation:

```
./configure --prefix=/usr --libexecdir=/usr/lib \
    --docdir=/usr/share/doc/man-db-2.6.7.1 --sysconfdir=/etc \
    --disable-setuid --with-browser=/usr/bin/lynx \
    --with-vgrind=/usr/bin/vgrind --with-grap=/usr/bin/grap
```

The meaning of the configure options:

--disable-setuid

This disables making the **man** program setuid to user `man`.

--with-...

These three parameters are used to set some default programs. **lynx** is a text-based web browser (see CBLFS for installation instructions), **vgrind** converts program sources to Groff input, and **grap** is useful for typesetting graphs in Groff documents. The **vgrind** and **grap** programs are not normally needed for viewing manual pages. They are not part of CLFS or CBLFS, but you should be able to install them yourself after finishing CLFS if you wish to do so.

Compile the package:

```
make
```

To test the results, issue:

```
make check
```

Install the package:

```
make install
```

10.51.2. Non-English Manual Pages in CLFS

The following table shows the character set that Man-DB assumes manual pages installed under `/usr/share/man/<ll>` will be encoded with. In addition to this, Man-DB correctly determines if manual pages installed in that directory are UTF-8 encoded.

Table 10.1. Expected character encoding of legacy 8-bit manual pages

Language (code)	Encoding	Language (code)	Encoding
Danish (da)	ISO-8859-1	Croatian (hr)	ISO-8859-2
German (de)	ISO-8859-1	Hungarian (hu)	ISO-8859-2
English (en)	ISO-8859-1	Japanese (ja)	EUC-JP
Spanish (es)	ISO-8859-1	Korean (ko)	EUC-KR
Estonian (et)	ISO-8859-1	Lithuanian (lt)	ISO-8859-13
Finnish (fi)	ISO-8859-1	Latvian (lv)	ISO-8859-13
French (fr)	ISO-8859-1	Macedonian (mk)	ISO-8859-5
Irish (ga)	ISO-8859-1	Polish (pl)	ISO-8859-2
Galician (gl)	ISO-8859-1	Romanian (ro)	ISO-8859-2
Indonesian (id)	ISO-8859-1	Russian (ru)	KOI8-R
Icelandic (is)	ISO-8859-1	Slovak (sk)	ISO-8859-2
Italian (it)	ISO-8859-1	Slovenian (sl)	ISO-8859-2
Norwegian Bokmal (nb)	ISO-8859-1	Serbian Latin (sr@latin)	ISO-8859-2
Dutch (nl)	ISO-8859-1	Serbian (sr)	ISO-8859-5
Norwegian Nynorsk (nn)	ISO-8859-1	Turkish (tr)	ISO-8859-9
Norwegian (no)	ISO-8859-1	Ukrainian (uk)	KOI8-U
Portuguese (pt)	ISO-8859-1	Vietnamese (vi)	TCVN5712-1
Swedish (sv)	ISO-8859-1	Simplified Chinese (zh_CN)	GBK
Belarusian (be)	CP1251	Simplified Chinese, Singapore (zh_SG)	GBK
Bulgarian (bg)	CP1251	Traditional Chinese, Hong Kong (zh_HK)	BIG5HKSCS
Czech (cs)	ISO-8859-2	Traditional Chinese (zh_TW)	BIG5
Greek (el)	ISO-8859-7		

Note

Manual pages in languages not in the list are not supported.

10.51.3. Contents of Man-DB

Installed programs: accessdb, apropos (link to whatis), catman, lexgrog, man, mandb, manpath, whatis, zsoelim

Installed libraries: libman.so, libmandb.so

Installed directories: /usr/lib/man-db, /usr/share/doc/man-db-2.6.7.1

Short Descriptions

accessdb	Dumps the **whatis** database contents in human-readable form
apropos	Searches the **whatis** database and displays the short descriptions of system commands that contain a given string
catman	Creates or updates the pre-formatted manual pages
lexgrog	Displays one-line summary information about a given manual page
man	Formats and displays the requested manual page
mandb	Creates or updates the **whatis** database
manpath	Displays the contents of $MANPATH or (if $MANPATH is not set) a suitable search path based on the settings in man.conf and the user's environment
whatis	Searches the **whatis** database and displays the short descriptions of system commands that contain the given keyword as a separate word
zsoelim	Reads files and replaces lines of the form *.so file* by the contents of the mentioned *file*
`libman`	Contains run-time support for **man**
`libmandb`	Contains run-time support for **man**

10.52. Make-4.0

The Make package contains a program for compiling packages.

10.52.1. 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
```

10.52.2. 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

10.53. XZ Utils-5.0.5

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

10.53.1. Installation of XZ Utils

Prepare XZ Utils for compilation:

```
./configure --prefix=/usr --docdir=/usr/share/doc/xz-5.0.5
```

Compile the package:

```
make
```

To test the results, issue:

```
make check
```

Install the programs:

```
make install
```

Move the `xz` binary, and several symlinks that point to it, into the /bin directory:

```
mv -v /usr/bin/{xz,lzma,lzcat,unlzma,unxz,xzcat} /bin
```

Finally, move the shared library to a more appropriate location, and recreate the symlink pointing to it:

```
mv -v /usr/lib/liblzma.so.* /lib
ln -sfv ../../lib/$(readlink /usr/lib/liblzma.so) /usr/lib/liblzma.so
```

10.53.2. Contents of XZ Utils

Installed programs:	lzcat (link to xz), lzcmp (link to xzdiff), lzdiff (link to xzdiff), lzegrep (link to xzgrep), lzfgrep (link to xzgrep), lzgrep (link to xzgrep), lzless (link to xzless), lzma (link to xz), lzmadec, lzmainfo, lzmore (link to xzmore), unlzma (link to xz), unxz (link to xz), xz, xzcat (link to xz), xzcmp (link to xzdiff), xzdec, xzdiff, xzegrep (link to xzgrep), xzfgrep (link to xzgrep), xzgrep, xzless, xzmore
Installed libraries:	liblzma.[a,so]
Installed directories:	/usr/include/lzma, /usr/share/doc/xz-5.0.5

Short Descriptions

lzcat	Decompresses LZMA and xz files
lzcmp	Compares lzma compressed files
lzdiff	Compares lzma compressed files
lzegrep	Runs **egrep** on lzma compressed files
lzfgrep	Runs **fgrep** on lzma compressed files
lzgrep	Runs **grep** on lzma compressed files
lzless	Runs **less** on lzma files

lzma	Compresses lzma files
lzmadec	Decompresses lzma files
lzmainfo	Displays information stored in an .lzma file header
lzmore	Runs **more** on lzma files
unlzma	Uncompresses lzma files
unxz	Uncompresses xz files
xz	Creates xz compressed files
xzcat	Decompresses xz files
xzcmp	Compares xz compressed files
xzdec	Decompresses to standard output
xzdiff	Compares xz compressed files
xzegrep	Runs **egrep** on xz compressed files
xzfgrep	Runs **fgrep** on xz compressed files
xzgrep	Runs **grep** on xz compressed files
xzless	Runs **less** on xz files
xzmore	Runs **more** on xz files
`liblzma`	The LZMA library

10.54. Kmod-18

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

10.54.1. Installation of Kmod

Prepare Kmod for compilation:

```
./configure --prefix=/usr \
    --bindir=/bin --sysconfdir=/etc \
    --with-rootlibdir=/lib \
    --with-zlib --with-xz
```

The meaning of the configure option:

--with-rootlibdir=/lib
: Install location for shared libraries.

--with-zlib --with-xz
: This allows the Kmod package to handle zlib and XZ compressed kernel modules.

Compile the package:

```
make
```

To test the results, issue:

```
make check
```

Install the package:

```
make install
```

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

```
ln -sfv kmod /bin/lsmod
for tool in depmod insmod modinfo modprobe rmmod; do
    ln -sfv ../bin/kmod /sbin/${tool}
done
```

10.54.2. Contents of Kmod

Installed programs: depmod (link to kmod), insmod (link to kmod), kmod, lsmod (link to kmod), modinfo (link to kmod), modprobe (link to kmod), rmmod (link to kmod)

Short Descriptions

depmod Creates a dependency file based on the symbols it finds in the existing set of modules; this dependency file is used by **modprobe** to automatically load the required modules

insmod Installs a loadable module in the running kernel

kmod Loads and unloads kernel modules

lsmod	Lists currently loaded modules
modinfo	Examines an object file associated with a kernel module and displays any information that it can glean
modprobe	Uses a dependency file, created by **depmod**, to automatically load relevant modules
rmmod	Unloads modules from the running kernel

10.55. Patch-2.7.1

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

10.55.1. Installation of Patch

Prepare Patch for compilation:

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

Compile the package:

```
make
```

To test the results, issue:

```
make check
```

Install the package:

```
make install
```

10.55.2. 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.

10.56. Psmisc-22.21

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

10.56.1. Installation of Psmisc

Prepare Psmisc for compilation:

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

Compile the package:

```
make
```

This package does not come with a test suite.

Install the package:

```
make install
```

10.56.2. Contents of Psmisc

Installed programs: fuser, killall, peekfd, prtstat, pstree, pstree.x11 (link to pstree)

Short Descriptions

fuser	Reports the Process IDs (PIDs) of processes that use the given files or file systems
killall	Kills processes by name; it sends a signal to all processes running any of the given commands
peekfd	Peeks at file descriptors of running processes
prtstat	Prints information about a process
pstree	Displays running processes as a tree
pstree.x11	Same as **pstree**, except that it waits for confirmation before exiting

10.57. Libestr-0.1.5

The Libestr package is a library for some string essentials.

10.57.1. Installation of Libestr

Prepare Libestr for compilation:

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

Compile the package:

```
make
```

This package does not come with a test suite.

Install the package:

```
make install
```

10.57.2. Contents of Libestr

Installed libraries: libestr.[a,so]

Short Descriptions

`libestr` contains functions for aiding in string functions

10.58. Libee-0.4.1

The Libee is an event expression library.

10.58.1. Installation of Libee

Prepare Libee for compilation:

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

Compile the package:

> **Note**
>
> Libee will fail to compile if using multiple jobs with make. Append "**-j 1**" to the following make command:

```
make
```

This package does not come with a test suite.

Install the package:

```
make install
```

10.58.2. Contents of Libee

Installed Program:	libee-convert
Installed libraries:	libee.[a,so]
Installed directory:	/usr/include/libee

Short Descriptions

libee-convert	todo
`libee`	is the event expression library

10.59. Rsyslog-6.4.2

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

10.59.1. Installation of Rsyslog

Prepare Rsyslog for compilation:

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

Compile the package:

```
make
```

To test the results, issue:

```
make check
```

Install the package:

```
make install
```

Create a directory for expansion snippets:

```
install -dv /etc/rsyslog.d
```

```
$ModLoad imklog.so

########################################################################
# Global Options

# Use traditional timestamp format.
$ActionFileDefaultTemplate RSYSLOG_TraditionalFileFormat

# Set the default permissions for all log files.
$FileOwner root
$FileGroup root
$FileCreateMode 0640
$DirCreateMode 0755

# Provides UDP reception
$ModLoad imudp
$UDPServerRun 514

# Disable Repeating of Entries
$RepeatedMsgReduction on

########################################################################
# Include Rsyslog Config Snippets

$IncludeConfig /etc/rsyslog.d/*.conf

########################################################################
# Standard Log Files

auth,authpriv.*   /var/log/auth.log
*.*;auth,authpriv.none  -/var/log/syslog
daemon.*   -/var/log/daemon.log
kern.*    -/var/log/kern.log
lpr.*    -/var/log/lpr.log
mail.*    -/var/log/mail.log
user.*    -/var/log/user.log

# Catch All Logs
*.=debug;\
 auth,authpriv.none;\
 news.none;mail.none -/var/log/debug
*.=info;*.=notice;*.=warn;\
 auth,authpriv.none;\
 cron,daemon.none;\
 mail,news.none  -/var/log/messages

# Emergencies are shown to everyone
*.emerg    *

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

10.59.3. Contents of rsyslog

Installed programs:	rsyslogd
Installed directory:	/usr/lib/rsyslog

Short Descriptions

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

10.60. Sysvinit-2.88dsf

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

10.60.1. Installation of Sysvinit

Apply a **sed** to disable several programs from being built and installed as better versions are provided by other packages:

```
sed -i -e 's/\ sulogin[^ ]*//' -e 's/pidof\.8//' -e '/ln .*pidof/d' \
    -e '/utmpdump/d' -e '/mountpoint/d' -e '/mesg/d' src/Makefile
```

Compile the package:

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

Install the package:

```
make -C src install
```

10.60.2. Configuring Sysvinit

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

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

id:3:initdefault:

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

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

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

su:S016:once:/sbin/sulogin

EOF
```

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

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

EOF
```

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

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

EOF
```

Finally, add the end line to `/etc/inittab`:

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

The *-I '\033(K'* option tells **agetty** to send this escape sequence to the terminal before doing anything else. This escape sequence switches the console character set to a user-defined one, which can be modified by running the **setfont** program. The **i18n** 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.

10.60.3. Contents of Sysvinit

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

Short Descriptions

bootlogd	Logs boot messages to a log file
fstab-decode	Runs a command with fstab-encoded arguments
halt	Normally invokes **shutdown** with the *-h* option, except when already in run-level 0, then it tells the kernel to halt the system; it notes in the file `/var/log/wtmp` that the system is being brought down
init	The first process to be started when the kernel has initialized the hardware which takes over the boot process and starts all the proceses it is instructed to
killall5	Sends a signal to all processes, except the processes in its own session so it will not kill the shell running the script that called it
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 `/run/utmp`
shutdown	Brings the system down in a secure way, signaling all processes and notifying all logged-in users
telinit	Tells **init** which run-level to change to

10.61. Tar-1.27.1

The Tar package contains an archiving program.

10.61.1. Installation of Tar

The following patch adds a man page for **tar**:

```
patch -Np1 -i ../tar-1.27.1-manpage-1.patch
```

Prepare Tar for compilation:

```
FORCE_UNSAFE_CONFIGURE=1 ./configure --prefix=/usr \
    --bindir=/bin --libexecdir=/usr/sbin
```

Compile the package:

```
make
```

To test the results, issue:

```
make check
```

Install the package:

```
make install
```

Generate the man page and place it in the proper location:

```
perl tarman > /usr/share/man/man1/tar.1
```

Install the documentation:

```
make -C doc install-html docdir=/usr/share/doc/tar-1.27.1
```

10.61.2. Contents of Tar

Installed programs: rmt, tar
Installed directory: /usr/share/doc/tar-1.27.1

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

10.62. Texinfo-5.2

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

10.62.1. 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
```

10.62.2. Contents of Texinfo

Installed programs: info, infokey, install-info, makeinfo (link to texi2any), pdftexi2dvi, texi2dvi, texi2pdf, texindex

Installed directory: /usr/share/texinfo

Short Descriptions

info	Used to read info pages which are similar to man pages, but often go much deeper than just explaining all the command line options. For example, compare **man bison** and **info bison**.
infokey	Compiles a source file containing Info customizations into a binary format
install-info	Used to install info pages; it updates entries in the **info** index file
makeinfo	Translates the given Texinfo source documents into info pages, plain text, or HTML
pdftexi2dvi	Shell script that run **texi2dvi --pdf**
texi2dvi	Used to format the given Texinfo document into a device-independent file that can be printed
texi2pdf	Used to format the given Texinfo document into a Portable Document Format (PDF) file
texindex	Used to sort Texinfo index files

10.63. Eudev-1.7

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

10.63.1. Installation of Eudev

Prepare Eudev for compilation:

```
./configure --prefix=/usr --sysconfdir=/etc \
    --with-rootprefix="" --libexecdir=/lib --enable-split-usr \
    --libdir=/usr/lib --with-rootlibdir=/lib --sbindir=/sbin --bindir=/sbin \
    --enable-rule_generator --disable-introspection --disable-keymap \
    --disable-gudev --disable-gtk-doc-html --with-firmware-path=/lib/firmware \
    --enable-libkmod
```

Compile the package:

```
make
```

To test the results, issue:

```
make check
```

Install the package:

```
make install
```

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

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

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

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

10.63.2. Contents of Eudev

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

Short Descriptions

udevadm	Controls the runtime behavior of Eudev, requests kernel events, manages the event queue, and provides simple debugging.
udevd	A daemon that reorders hotplug events before submitting them to **udev**, thus avoiding various race conditions
ata_id	Provides Eudev with a unique string and additional information (uuid, label) for an ATA drive
cdrom_id	Prints the capabilities of a CDROM or DVDROM drive.

collect	Given an ID for the current uevent and a list of IDs (for all target uevents), registers the current ID and indicates whether all target IDs have been registered.
create_floppy_devices	Creates all possible floppy devices based on the CMOS type
edd_id	Identifies x86 disk drives from Enhanced Disk Drive calls
firmware.sh	Script to load firmware for a device
fstab_import	Finds an entry in `/etc/fstab` that matches the current device, and provides its information to Udev.
path_id	Provides the shortest possible unique hardware path to a device
scsi_id	Retrieves or generates a unique SCSI identifier.
usb_id	Identifies a USB block device.
v4l_id	Determines V4L capabilities for a given device.
write_cd_rules	A script which generates Eudev rules to provide stable names for network interfaces.
write_net_rules	A script which generates Eudev rules to provide stable names for network interfaces.
`libudev`	A library interface to eudev device information.
`/etc/udev`	Contains **udev** configuration files, device permissions, and rules for device naming
`/lib/udev`	Contains **udev** helper programs and static devices which get copied to /dev when booted.

10.64. Vim-7.4

The Vim package contains a powerful text editor.

10.64.1. Installation of Vim

Alternatives to Vim

If you prefer another editor—such as Emacs, Joe, or Nano—please refer to *http://cblfs.cross-lfs.org/index.php/Category:Text_Editors* for suggested installation instructions.

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

```
patch -Np1 -i ../vim-7.4-branch_update-7.patch
```

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

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

Prepare Vim for compilation:

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

Compile the package:

```
make
```

To test the results, issue:

```
make test
```

However, this test suite outputs a lot of binary data to the screen, which can cause issues with the settings of the current terminal. This can be resolved by redirecting the output to a log file.

Install the package:

```
make install
```

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

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

By default, Vim's documentation is installed in `/usr/share/vim`. The following symlink allows the documentation to be accessed via `/usr/share/doc/vim-7.4`, making it consistent with the location of documentation for other packages:

```
ln -sv ../vim/vim74/doc /usr/share/doc/vim-7.4
```

If an X Window System is going to be installed on the CLFS system, you may want to recompile Vim after installing X. Vim comes with a GUI version of the editor that requires X and some additional libraries to be installed. For more information, refer to the Vim documentation and the Vim installation page in CBLFS at *http://cblfs.cross-lfs.org/index.php/Vim*.

10.64.2. Configuring Vim

By default, **vim** runs in vi-incompatible mode. This may be new to users who have used other editors in the past. The “nocompatible” setting is included below to highlight the fact that a new behavior is being used. It also reminds those who would change to “compatible” mode that it should be the first setting in the configuration file. This is necessary because it changes other settings, and overrides must come after this setting. Create a default **vim** configuration file by running the following:

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

set nocompatible
set backspace=2
set ruler
syntax on
if (&term == "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'
```

10.64.3. 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, xxd

Installed directory: /usr/share/vim

Short Descriptions

efm_filter.pl A filter for creating an error file that can be read by **vim**

efm_perl.pl Reformats the error messages of the Perl interpreter for use with the “quickfix” mode of **vim**

ex Starts **vim** in ex mode

less.sh A script that starts **vim** with less.vim

mve.awk Processes **vim** errors

pltags.pl Creates a tags file for Perl code for use by **vim**

ref Checks the spelling of arguments

rview	Is a restricted version of **view**; no shell commands can be started and **view** cannot be suspended
rvim	Is a restricted version of **vim**; no shell commands can be started and **vim** cannot be suspended
shtags.pl	Generates a tags file for Perl scripts
tcltags	Generates a tags file for TCL code
view	Starts **vim** in read-only mode
vi	Link to **vim**
vim	Is the editor
vim132	Starts **vim** with the terminal in 132-column mode
vim2html.pl	Converts Vim documentation to HypterText Markup Language (HTML)
vimdiff	Edits two or three versions of a file with **vim** and show differences
vimm	Enables the DEC locator input model on a remote terminal
vimspell.sh	Spell checks a file and generates the syntax statements necessary to highlight in **vim**. This script requires the old Unix **spell** command, which is provided neither in CLFS nor in CBLFS
vimtutor	Teaches the basic keys and commands of **vim**
xxd	Creates a hex dump of the given file; it can also do the reverse, so it can be used for binary patching

10.65. Hfsutils-3.2.6

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

10.65.1. Installation of Hfsutils

Apply the following patch to add a missing `errno.h` include and allow HFSutils to recognize devices larger than 2GB:

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

Prepare Hfsutils for compilation:

```
./configure --prefix=/usr --mandir=/usr/share/man
```

Compile the package:

```
make
```

Install the package:

```
make install
```

10.65.2. Contents of Hfsutils

Installed programs: hattrib, hcd, hcopy, hdel, hdir, hfsutils, hformat, hls, hmkdir, hmount, hpwd, hrename, hrmdir, humount, hvol (these are all hardlinks to hfsutils).

Short Descriptions

hattrib	Change FS file or directory attributes.
hcd	Change working HFS directory.
hcopy	Copy files to or from an HFS volume.
hdel	Delete both forks of an HFS file.
hdir	Display an HFS directory in long format.
hformat	Create a new HFS filesystem and make it current.
hfsutils	Tools for accessing Macintosh HFS-formatted volumes.
hls	List files in an HFS directory.
hmkdir	Create a new HFS directory.
hmount	Introduce a new HFS volume and make it current.
hpwd	Print the full path to the current HFS working directory.
hrename	Rename or move an HFS file or directory.
hrmdir	Remaove an empty HFS directory.
humount	Remove an HFS volume from the list of known volumes.
hvol	Display or change the current HFS volume.

10.66. Parted-3.1

Parted is a program for creating, copying and modifying partitions, and the file systems on them. Parted is especially useful on PPC machines in that, unlike **fdisk**, it accurately reads Macintosh partition maps.

10.66.1. Installation of Parted

Prepare Parted for compilation:

```
./configure --prefix=/usr --disable-device-mapper
```

The meaning of the configure options:

--disable-device-mapper
: This disables the use of the device-mapper library, which we do not install in CLFS.

Compile the Parted package:

```
make
```

Install the package:

```
make install
```

10.66.2. Contents of Parted

Installed programs:	parted, partprobe
Installed libraries:	libparted.[a,so]

Short Descriptions

parted	A program for creating, destroying, resizing, checking and copying partitions, and the filesystems on them. This is useful for creating space for new operating systems, reorganising disk usage, copying data between hard disks, and disk imaging.
partprobe	Informs the OS of partition table changes.
`libparted`	A library to manipulate partitions.

10.67. Powerpc-Utils_1.1.3

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

10.67.1. Installation of Powerpc-Utils

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

```
patch -Np1 -i ../powerpc-utils_1.1.3-fixes-2.patch
```

Compile the needed programs:

```
make nvsetenv nvsetvol
```

Install the package:

```
install -v -m755 nvsetenv nvsetvol /usr/sbin
install -v -m644 nvsetenv.8 nvsetvol.8 /usr/share/man/man8
```

10.67.2. Contents of Powerpc-Utils

Installed programs: nvsetenv, nvsetvol

Short Descriptions

nvsetenv Manipulate variables in the non-volatile RAM.

nvsetvol Adjust the volume of the boot-up chime on Macintoshes.

10.68. Yaboot-1.3.17

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

10.68.1. Installation of Yaboot

The following patch adds stub functions for newer e2fsprogs releases:

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

The following patch adds Parted support to yabootconfig:

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

The supplied man pages have `/usr/local` in the text. This sed will correct that:

```
sed -i 's%/usr/local%/usr%' man/*
```

Prevent the build from failing due to warnings with the following sed:

```
sed -i 's%-Werror%%' Makefile
```

Compile the package:

```
make PREFIX=/usr
```

Install the package:

```
make PREFIX=/usr install
```

10.68.2. Contents of Yaboot

Installed programs:	addnote, mkofboot (link to ybin), ofboot, ofpath, yaboot, yabootconfig, ybin
Installed files:	yaboot.conf

Short Descriptions

addnote	For IBM CHRP machines, add a PT_NOTE program header entry to an elf file so that it can be booted.
mkofboot	Format the bootstrap partition and install the yaboot boot loader.
ofboot	Script to format the boot menu using yaboot.conf and write the resulting Open Firmware code to the bootstrap.
ofpath	Determine Open Firmware path corresponding to a device node.
yaboot	Open Firmware boot loader.
yabootconfig	Generate and install a simple yaboot.conf.
ybin	Shell script to update or install the boot loader on a bootstrap partition.
`yaboot.conf`	Configuration file used by **ybin** to determine how to install **yaboot** on the bootstrap partition.

10.69. 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.

10.70. Stripping

If the intended user is not a programmer and does not plan to do any debugging on the system software, the system size can be decreased by about 200 MB by removing the debugging symbols from binaries and libraries. This causes no inconvenience other than not being able to debug the software fully anymore.

Most people who use the command mentioned below do not experience any difficulties. However, it is easy to make a typo and render the new system unusable, so before running the **strip** command, it is a good idea to make a backup of the current situation.

Before performing the stripping, take special care to ensure that none of the binaries that are about to be stripped are running. If unsure whether the user entered chroot with the command given in If You Are Going to Chroot first exit from chroot:

```
logout
```

Then reenter it with:

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

Now the binaries and libraries can be safely stripped:

```
/tools/bin/find /{,usr/}{bin,lib,sbin} -type f \
   -exec /tools/bin/strip --strip-debug '{}' ';'
```

A large number of files will be reported as having their file format not recognized. These warnings can be safely ignored. These warnings indicate that those files are scripts instead of binaries.

If disk space is very tight, the `--strip-all` option can be used on the binaries in `/{,usr/}{bin,sbin}` to gain several more megabytes. Do not use this option on libraries—they will be destroyed.

Chapter 11. System Configuration

11.1. Introduction

This chapter details how to install and configure the CLFS-Bootscripts package and customize other configuration files on the system. Most of the Bootscripts will work without modification, but a few require additional configuration files because they deal with hardware-dependent information.

System-V style init scripts are employed in this book because they are widely used. For additional options, a hint detailing the BSD style init setup is available at *http://hints.cross-lfs.org/index.php/BSD-Init*. Searching the LFS mailing lists for “depinit” will also offer additional choices.

If using an alternative style of init scripts, skip this chapter and move on to Making the CLFS System Bootable.

11.2. Bootscripts for CLFS 3.0-20140710

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

11.2.1. Installation of Bootscripts

Install the bootscripts:

```
make install-bootscripts
```

If you are going to use a network card, run the following command to install scripts for starting and configuring the network interface:

```
make install-network
```

11.2.2. Contents of Bootscripts

Installed scripts: checkfs, cleanfs, console, functions, halt, ifdown, ifup, localnet, mountfs, mountkernfs, network, rc, reboot, sendsignals, setclock, static, swap, sysklogd, template, and udev.

Short Descriptions

checkfs Checks the integrity of the file systems before they are mounted (with the exception of journal and network based file systems)

cleanfs Removes files that should not be preserved between reboots, such as those in `/run/` and `/var/lock/`; it re-creates `/run/utmp` and removes the possibly present `/etc/nologin`, `/fastboot`, and `/forcefsck` files

console Loads the correct keymap table for the desired keyboard layout; it also sets the screen font

functions Contains common functions, such as error and status checking, that are used by several bootscripts

halt Halts the system

ifdown 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 *noauto* or are network based

mountkernfs Mounts virtual kernel file systems, such as `proc`

network Sets up network interfaces, such as network cards, and sets up the default gateway (where applicable)

rc The master run-level control script; it is responsible for running all the other bootscripts one-by-one, in a sequence determined by the name of the symbolic links being processed

reboot Reboots the system

sendsignals Makes sure every process is terminated before the system reboots or halts

setclock Resets the kernel clock to local time in case the hardware clock is not set to UTC time

static Provides the functionality needed to assign a static Internet Protocol (IP) address to a network interface

swap Enables and disables swap files and partitions

sysklogd	Starts and stops the system and kernel log daemons
template	A template to create custom bootscripts for other daemons
udev	Starts and stops the Eudev daemon

11.3. How Do These Bootscripts Work?

Linux uses a special booting facility named SysVinit that is based on a concept of *run-levels*. It can be quite different from one system to another, so it cannot be assumed that because things worked in one particular Linux distribution, they should work the same in CLFS too. CLFS has its own way of doing things, but it respects generally accepted standards.

SysVinit (which will be referred to as “init” from now on) works using a run-levels scheme. There are seven (numbered 0 to 6) run-levels (actually, there are more run-levels, but they are for special cases and are generally not used. See `init(8)` for more details), and each one of those corresponds to the actions the computer is supposed to perform when it starts up. The default run-level is 3. Here are the descriptions of the different run-levels as they are implemented:

```
0: halt the computer
1: single-user mode
2: multi-user mode without networking
3: multi-user mode with networking
4: reserved for customization, otherwise does the same as 3
5: same as 4, it is usually used for GUI login (like X's xdm or KDE's kdm)
6: reboot the computer
```

The command used to change run-levels is **init *`[runlevel]`***, where *`[runlevel]`* is the target run-level. For example, to reboot the computer, a user could issue the **init 6** command, which is an alias for the **reboot** command. Likewise, **init 0** is an alias for the **halt** command.

There are a number of directories under `/etc/rc.d` that look like `rc?.d` (where ? is the number of the run-level) and `rcsysinit.d`, all containing a number of symbolic links. Some begin with a *K*, the others begin with an *S*, and all of them have two numbers following the initial letter. The K means to stop (kill) a service and the S means to start a service. The numbers determine the order in which the scripts are run, from 00 to 99—the lower the number the earlier it gets executed. When **init** switches to another run-level, the appropriate services are either started or stopped, depending on the runlevel chosen.

The real scripts are in `/etc/rc.d/init.d`. They do the actual work, and the symlinks all point to them. Killing links and starting links point to the same script in `/etc/rc.d/init.d`. This is because the scripts can be called with different parameters like `start`, `stop`, `restart`, `reload`, and `status`. When a K link is encountered, the appropriate script is run with the `stop` argument. When an S link is encountered, the appropriate script is run with the `start` argument.

There is one exception to this explanation. Links that start with an *S* in the `rc0.d` and `rc6.d` directories will not cause anything to be started. They will be called with the parameter `stop` to stop something. The logic behind this is that when a user is going to reboot or halt the system, nothing needs to be started. The system only needs to be stopped.

These are descriptions of what the arguments make the scripts do:

`start`
: The service is started.

`stop`
: The service is stopped.

`restart`
: The service is stopped and then started again.

`reload`
: The configuration of the service is updated. This is used after the configuration file of a service was modified, when the service does not need to be restarted.

`status`
: Tells if the service is running and with which PIDs.

Feel free to modify the way the boot process works (after all, it is your own CLFS system). The files given here are an example of how it can be done.

11.4. Configuring the setclock Script

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

If you cannot remember whether or not the hardware clock is set to UTC, find out by running the **`hwclock --localtime --show`** command. This will display what the current time is according to the hardware clock. If this time matches whatever your watch says, then the hardware clock is set to local time. If the output from **hwclock** is not local time, chances are it is set to UTC time. Verify this by adding or subtracting the proper amount of hours for the timezone to the time shown by **hwclock**. For example, if you are currently in the MST timezone, which is also known as GMT -0700, add seven hours to the local time.

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

Create a new file `/etc/sysconfig/clock` by running the following:

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

UTC=1

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

A good hint explaining how to deal with time on CLFS is available at *http://hints.cross-lfs.org/index.php/time.txt*. It explains issues such as time zones, UTC, and the `TZ` environment variable.

11.5. Configuring the Linux Console

This section discusses how to configure the **i18n** bootscript that sets up the keyboard map and the console font. If non-ASCII characters (e.g., the British pound sign and Euro character) will not be used and the keyboard is a U.S. one, skip this section. Without the configuration file, the **i18n** bootscript will do nothing.

The **i18n** script reads the `/etc/sysconfig/i18n` file for configuration information. Decide which keymap and screen font will be used. Various language-specific HOWTO's can also help with this (see *http://www.tldp.org/HOWTO/HOWTO-INDEX/other-lang.html*. A pre-made `/etc/sysconfig/i18n` file with known settings for several countries was installed with the CLFS-Bootscripts package, so the relevant section can be uncommented if the country is supported. If still in doubt, look in the `/usr/share/consolefonts` for valid screen fonts and `/usr/share/keymaps` for valid keymaps.

The default `/etc/sysconfig/i18n` is set up for UTF-8 using the us keymap. You will need to edit the file to your specific needs. The `/etc/sysconfig/i18n` file has additional information in it to help you to assist in configuring.

11.6. Device and Module Handling on a CLFS System

In Installing Basic System Software, we installed the Eudev package. Before we go into the details regarding how this works, a brief history of previous methods of handling devices is in order.

11.6.1. History

11.6.1.1. Static Device Nodes

Linux systems in general traditionally use a static device creation method, whereby a great many device nodes are created under `/dev` (sometimes literally thousands of nodes), regardless of whether the corresponding hardware devices actually exist. This is typically done via a **MAKEDEV** script, which contains a number of calls to the **mknod** program with the relevant major and minor device numbers for every possible device that might exist in the world.

11.6.1.2. Devfs

In February 2000, a new filesystem called `devfs`, which dynamically created device nodes as devices were found by the kernel, was merged into the 2.3.46 kernel and was made available during the 2.4 series of stable kernels. Although it was present in the kernel source itself, this method of creating devices dynamically never received overwhelming support from the core kernel developers.

The main problem with the approach adopted by `devfs` was the way it handled device detection, creation, and naming. The latter issue, that of device node naming, was perhaps the most critical. It is generally accepted that if device names are allowed to be configurable, then the device naming policy should be up to a system administrator, not imposed on them by any particular developer(s). The `devfs` file system also suffered from race conditions that were inherent in its design and could not be fixed without a substantial revision to the kernel. It was marked deprecated with the release of the 2.6 kernel series, and was removed entirely as of version 2.6.18.

11.6.1.3. Sysfs

With the development of the unstable 2.5 kernel tree, later released as the 2.6 series of stable kernels, a new virtual filesystem called `sysfs` came to be. The job of `sysfs` is to export a view of the system's hardware configuration to userspace processes. Drivers that have been compiled into the kernel directly register their objects with `sysfs` as they are detected by the kernel. For drivers compiled as modules, this registration will happen when the module is loaded. Once the `sysfs` filesystem is mounted (on `/sys`), data which the built-in drivers registered with `sysfs` are available to userspace processes. With this userspace-visible representation, the possibility of seeing a userspace replacement for `devfs` became much more realistic.

11.6.1.4. Udev Implementation

Shortly after the introduction of `sysfs`, work began on a program called Udev to advantage of it. The **udev** daemon made calls to `mknod()` to create device nodes in `/dev` dynamically, based on the information from `sysfs`, in `/sys`. For example, `/sys/class/tty/vcs/dev` contains the string “7:0”. This string was used by **udev** to create a device node with major number *7* and minor number *0*.

Linux kernel version 2.6.32 introduced a new virtual file system called `devtmpfs`, an improved replacement for `devfs`. This allows device nodes to once again be dynamically created by the kernel, without many of the problems of `devfs`. As of version 176, Udev no longer creates device nodes itself, instead relying on `devtmpfs` to do so.

11.6.1.5. Systemd and Eudev

In 2010, development began on systemd, an alternate **init** implementation. Starting with Udev 183, Udev's source tree was merged with systemd. Several Gentoo developers who disagreed with this merge announced a project fork called Eudev in December 2012, created by extracting the Udev code from systemd. One of the goals of Eudev is to allow for easier installation and usage of **udevd** without the need for the rest of systemd.

11.6.2. Device Node Creation

By default, device nodes created by the kernel in a `devtmpfs` are owned by *root:root* and have *600* permissions. **udevd** can modify ownership and permissions of the nodes under the `/dev` directory, and can also create additional symlinks, based on rules specified in the files within the `/etc/udev/rules.d`, `/lib/udev/rules.d`, and `/run/udev/rules.d` directories. The names for these files start with a number, to indicate the order in which they are run, and they have a `.rules` extension (**udevd** will ignore files with any other extension). All of the rules files from these directories are combined into a single list, sorted by filename, and run in that order. In the event of a conflict, where a rules file with the same name exists in two or more of these directories, the rules in `/etc` take the highest priority, followed by rules files in `/run`, and finally `/lib`. Any device for which a rule cannot be found will just be ignored by **udevd** and be left at the defaults defined by the kernel, as described above.

11.6.3. Module Loading

Device drivers compiled as modules may have aliases built into them. Aliases are visible in the output of the **modinfo** program and are usually related to the bus-specific identifiers of devices supported by a module. For example, the *snd-fm801* driver supports PCI devices with vendor ID 0x1319 and device ID 0x0801, and has an alias of “pci:v00001319d00000801sv*sd*bc04sc01i*”. For most devices, the bus driver exports the alias of the driver that would handle the device via `sysfs`. E.g., the `/sys/bus/pci/devices/0000:00:0d.0/modalias` file might contain the string “pci:v00001319d00000801sv00001319sd00001319bc04sc01i00”. The default rules provided by Eudev will cause **udevd** to call out to **/sbin/modprobe** with the contents of the `MODALIAS` uevent environment variable (that should be the same as the contents of the `modalias` file in sysfs), thus loading all modules whose aliases match this string after wildcard expansion.

In this example, this means that, in addition to *snd-fm801*, the obsolete (and unwanted) *forte* driver will be loaded if it is available. See below for ways in which the loading of unwanted drivers can be prevented.

The kernel itself is also able to load modules for network protocols, filesystems and NLS support on demand.

11.6.4. Problems with Loading Modules and Creating Devices

There are a few possible problems when it comes to automatically creating device nodes.

11.6.4.1. A kernel module is not loaded automatically

Eudev will only load a module if it has a bus-specific alias and the bus driver properly exports the necessary aliases to `sysfs`. In other cases, one should arrange module loading by other means. With Linux-3.14.21, Eudev is known to load properly-written drivers for INPUT, IDE, PCI, USB, SCSI, SERIO and FireWire devices.

To determine if the device driver you require has the necessary support for Eudev, run **modinfo** with the module name as the argument. Now try locating the device directory under `/sys/bus` and check whether there is a `modalias` file there.

If the `modalias` file exists in `sysfs`, the driver supports the device and can talk to it directly, but doesn't have the alias, it is a bug in the driver. Load the driver without the help from Eudev and expect the issue to be fixed later.

If there is no `modalias` file in the relevant directory under `/sys/bus`, this means that the kernel developers have not yet added modalias support to this bus type. With Linux-3.14.21, this is the case with ISA busses. Expect this issue to be fixed in later kernel versions.

Eudev is not intended to load “wrapper” drivers such as *snd-pcm-oss* and non-hardware drivers such as *loop* at all.

11.6.4.2. A kernel module is not loaded automatically, and Eudev is not intended to load it

If the “wrapper” module only enhances the functionality provided by some other module (e.g., *snd-pcm-oss* enhances the functionality of *snd-pcm* by making the sound cards available to OSS applications), configure **modprobe** to load the wrapper after Eudev loads the wrapped module. To do this, add an “install” line to a file in `/etc/modprobe.d`. For example:

```
install snd-pcm /sbin/modprobe -i snd-pcm ; \
    /sbin/modprobe snd-pcm-oss ; true
```

If the module in question is not a wrapper and is useful by itself, configure the **S05modules** bootscript to load this module on system boot. To do this, add the module name to the `/etc/sysconfig/modules` file on a separate line. This works for wrapper modules too, but is suboptimal in that case.

11.6.4.3. Eudev loads some unwanted module

Either don't build the module, or blacklist it in `/etc/modprobe.d` file as done with the *forte* module in the example below:

```
blacklist forte
```

Blacklisted modules can still be loaded manually with the explicit **modprobe** command.

11.6.4.4. Eudev makes a wrong symlink

This usually happens if a rule unexpectedly matches a device. For example, a poorly-written rule can match both a SCSI disk (as desired) and the corresponding SCSI generic device (incorrectly) by vendor. Find the offending rule and make it more specific, with the help of **udevadm info**.

11.6.4.5. Eudev rule works unreliably

This may be another manifestation of the previous problem. If not, and your rule uses `sysfs` attributes, it may be a kernel timing issue, to be fixed in later kernels. For now, you can work around it by creating a rule that waits for the used `sysfs` attribute and appending it to the `/etc/udev/rules.d/10-wait_for_sysfs.rules` file. Please notify the CLFS Development list if you do so and it helps.

11.6.4.6. Device naming order changes randomly after rebooting

This is due to the fact that Eudev, by design, handles uevents and loads modules in parallel, and thus in an unpredictable order. This will never be “fixed”. You should not rely upon the kernel device names being stable. Instead, create your own rules that make symlinks with stable names based on some stable attributes of the device, such as a serial number or the output of various *_id utilities installed by Eudev. See Section 11.7, “Creating custom symlinks to devices” and Networking Configuration for examples.

11.6.5. Useful Reading

Additional helpful documentation is available at the following sites:

- A Userspace Implementation of `devfs`

http://www.kroah.com/linux/talks/ols_2003_udev_paper/Reprint-Kroah-Hartman-OLS2003.pdf

- The `sysfs` Filesystem
  http://www.kernel.org/pub/linux/kernel/people/mochel/doc/papers/ols-2005/mochel.pdf

11.7. Creating custom symlinks to devices

11.7.1. CD-ROM symlinks

Some software that you may want to install later (e.g., various media players) expect the /dev/cdrom and /dev/dvd symlinks to exist. Also, it may be convenient to put references to those symlinks into `/etc/fstab`. For each of your CD-ROM devices, find the corresponding directory under `/sys` (e.g., this can be `/sys/block/hdd`) and run a command similar to the following:

```
udevadm test /sys/block/hdd
```

Look at the lines containing the output of various *_id programs.

There are two approaches to creating symlinks. The first one is to use the model name and the serial number, the second one is based on the location of the device on the bus. If you are going to use the first approach, create a file similar to the following:

```
cat >/etc/udev/rules.d/82-cdrom.rules << EOF

# Custom CD-ROM symlinks
SUBSYSTEM=="block", ENV{ID_MODEL}=="SAMSUNG_CD-ROM_SC-148F", \
    ENV{ID_REVISION}=="PS05", SYMLINK+="cdrom"
SUBSYSTEM=="block", ENV{ID_MODEL}=="PHILIPS_CDD5301", \
    ENV{ID_SERIAL}=="5VO1306DM00190", SYMLINK+="cdrom1 dvd"

EOF
```

Note

Although the examples in this book work properly, be aware that Eudev does not recognize the backslash for line continuation. If modifying Eudev rules with an editor, be sure to leave each rule on one physical line.

This way, the symlinks will stay correct even if you move the drives to different positions on the IDE bus, but the `/dev/cdrom` symlink won't be created if you replace the old SAMSUNG CD-ROM with a new drive.

The SUBSYSTEM=="block" key is needed in order to avoid matching SCSI generic devices. Without it, in the case with SCSI CD-ROMs, the symlinks will sometimes point to the correct `/dev/srX` devices, and sometimes to `/dev/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.

11.7.2. Dealing with duplicate devices

As explained in Section 11.6, "Device and Module Handling on a CLFS System", the order in which devices with the same function appear in `/dev` is essentially random. E.g., if you have a USB web camera and a TV tuner, sometimes `/dev/video0` refers to the camera and `/dev/video1` refers to the tuner, and sometimes after a reboot the order changes to the opposite one. For all classes of hardware except sound cards and network cards, this is fixable by creating udev rules for custom persistent symlinks. The case of network cards is covered separately in Networking Configuration, and sound card configuration can be found in *CBLFS*.

For each of your devices that is likely to have this problem (even if the problem doesn't exist in your current Linux distribution), find the corresponding directory under `/sys/class` or `/sys/block`. For video devices, this may be `/sys/class/video4linux/videoX`. Figure out the attributes that identify the device uniquely (usually, vendor and product IDs and/or serial numbers work):

```
udevadm info -a -p /sys/class/video4linux/video0
```

Then write rules that create the symlinks, e.g.:

```
cat >/etc/udev/rules.d/83-duplicate_devs.rules << EOF

# Persistent symlinks for webcam and tuner
KERNEL=="video*", SYSFS{idProduct}=="1910", SYSFS{idVendor}=="0d81", \
    SYMLINK+="webcam"
KERNEL=="video*", SYSFS{device}=="0x036f", SYSFS{vendor}=="0x109e", \
    SYMLINK+="tvtuner"

EOF
```

The result is that `/dev/video0` and `/dev/video1` devices still refer randomly to the tuner and the web camera (and thus should never be used directly), but there are symlinks `/dev/tvtuner` and `/dev/webcam` that always point to the correct device.

More information on writing Eudev rules can be found in `/usr/share/doc/udev/writing_udev_rules/index.html`.

11.8. The Bash Shell Startup Files

The shell program **/bin/bash** (hereafter referred to as “the shell”) uses a collection of startup files to help create an environment to run in. Each file has a specific use and may affect login and interactive environments differently. The files in the `/etc` directory provide global settings. If an equivalent file exists in the home directory, it may override the global settings.

An interactive login shell is started after a successful login, using **/bin/login**, by reading the `/etc/passwd` file. An interactive non-login shell is started at the command-line (e.g., `[prompt]$`**/bin/bash**). A non-interactive shell is usually present when a shell script is running. It is non-interactive because it is processing a script and not waiting for user input between commands.

For more information, see **info bash** under the *Bash Startup Files and Interactive Shells* section, and *Bash Startup Files* in CBLFS.

The files `/etc/profile` and `~/.bash_profile` are read when the shell is invoked as an interactive login shell. Create a base `/etc/profile` that will load any Bash auto completion files that may be on the system, and set the `INPUTRC` environment variable that makes Bash and Readline use `/etc/inputrc`:

```
cat > /etc/profile << "EOF"
# Begin /etc/profile

for f in /etc/bash_completion.d/*
do
  if [ -e ${f} ]; then source ${f}; fi
done
unset f

export INPUTRC=/etc/inputrc
EOF
```

11.9. Setting Up Locale Information

The instructions below explain how to add some environment variables necessary for native language support to the base `/etc/profile` created in the previous section. Setting these variables properly results in:

- The output of programs translated into the native language
- Correct classification of characters into letters, digits and other classes. This is necessary for **bash** to properly accept non-ASCII characters in command lines in non-English locales
- The correct alphabetical sorting order for the country
- Appropriate default paper size
- Correct formatting of monetary, time, and date values

Replace *`[ll]`* below with the two-letter code for the desired language (e.g., “en”) and *`[CC]`* with the two-letter code for the appropriate country (e.g., “GB”). *`[charmap]`* should be replaced with the canonical charmap for your chosen locale.

The list of all locales supported by Glibc can be obtained by running the following command:

```
locale -a
```

Locales can have a number of synonyms, e.g. “ISO-8859-1” is also referred to as “iso8859-1” and “iso88591”. Some applications cannot handle the various synonyms correctly, so it is safest to choose the canonical name for a particular locale. To determine the canonical name, run the following command, where *[locale name]* is the output given by **locale -a** for your preferred locale (“en_US.utf8” in our example).

```
LC_ALL=[locale name] locale charmap
```

For the “en_US.utf8” locale, the above command will print:

```
UTF-8
```

This results in a final locale setting of “en_US.UTF-8”. It is important that the locale found using the heuristic above is tested prior to it being added to the Bash startup files:

```
LC_ALL=[locale name] locale territory
LC_ALL=[locale name] locale language
LC_ALL=[locale name] locale charmap
LC_ALL=[locale name] locale int_curr_symbol
LC_ALL=[locale name] locale int_prefix
```

The above commands should print the language name, the character encoding used by the locale, the local currency, and the prefix to dial before the telephone number in order to get into the country. If any of the commands above fail with a message similar to the one shown below, this means that your locale was either not installed in Chapter 10 or is not supported by the default installation of Glibc.

```
locale: Cannot set LC_* to default locale: No such file or directory
```

If this happens, you should either install the desired locale using the **localedef** command, or consider choosing a different locale. Further instructions assume that there are no such error messages from Glibc.

Some packages beyond CLFS may also lack support for your chosen locale. One example is the X library (part of the X Window System), which outputs the following error message:

```
Warning: locale not supported by Xlib, locale set to C
```

Sometimes it is possible to fix this by removing the charmap part of the locale specification, as long as that does not change the character map that Glibc associates with the locale (this can be checked by running the **locale charmap** command in both locales). For example, one would have to change "de_DE.ISO-8859-15@euro" to "de_DE@euro" in order to get this locale recognized by Xlib.

Other packages can also function incorrectly (but may not necessarily display any error messages) if the locale name does not meet their expectations. In those cases, investigating how other Linux distributions support your locale might provide some useful information.

Once the proper locale settings have been determined, add them to the `/etc/profile` file:

```
cat >> /etc/profile << "EOF"

export LANG=[ll]_[CC].[charmap]

# End /etc/profile
EOF
```

Setting the keyboard layout, screen font, and locale-related environment variables are the only internationalization steps needed to support locales that use ordinary single-byte encodings and left-to-right writing direction. UTF-8 has been tested on the English, French, German, Italian, and Spanish locales. All other locales are untested. If you discover issues with any other locale please open a ticket in our Trac system.

Some locales need additional programs and support. CLFS will not be supporting these locales in the book. We welcome the support for these other locales via *http://cblfs.cross-lfs.org/*.

11.10. Creating the /etc/inputrc File

The `/etc/inputrc` file deals with mapping the keyboard for specific situations. This file is the start-up file used by Readline — the input-related library — used by Bash and most other shells.

Most people do not need user-specific keyboard mappings so the command below creates a global `/etc/inputrc` used by everyone who logs in. If you later decide you need to override the defaults on a per-user basis, you can create a `.inputrc` file in the user's home directory with the modified mappings.

For more information on how to edit the `inputrc` file, see **info bash** under the *Readline Init File* section. **info readline** is also a good source of information.

Below is a generic global `inputrc` along with comments to explain what the various options do. Note that comments cannot be on the same line as commands. Create the file using the following command:

```
cat > /etc/inputrc << "EOF"
# Begin /etc/inputrc
# Modified by Chris Lynn <roryo@roryo.dynup.net>

# Allow the command prompt to wrap to the next line
set horizontal-scroll-mode Off

# Enable 8bit input
set meta-flag On
set input-meta On

# Turns off 8th bit stripping
set convert-meta Off

# Keep the 8th bit for display
set output-meta On

# none, visible or audible
set bell-style none

# All of the following map the escape sequence of the
# value contained inside the 1st argument to the
# readline specific functions

"\eOd": backward-word
"\eOc": forward-word
```

```
# for linux console
"\e[1~": beginning-of-line
"\e[4~": end-of-line
"\e[5~": beginning-of-history
"\e[6~": end-of-history
"\e[3~": delete-char
"\e[2~": quoted-insert

# for xterm
"\eOH": beginning-of-line
"\eOF": end-of-line

# for Konsole
"\e[H": beginning-of-line
"\e[F": end-of-line

# End /etc/inputrc
EOF
```

11.11. 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
devpts         /dev/pts     devpts gid=5,mode=620   0     0
shm            /dev/shm     tmpfs  defaults         0     0

# End /etc/fstab
EOF
```

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

Chapter 12. Networking Configuration

12.1. Configuring the localnet Script

Part of the job of the **localnet** script is setting the system's hostname. This needs to be configured in the `/etc/sysconfig/network` file.

Create the `/etc/sysconfig/network` file and enter a hostname by running:

```
echo "HOSTNAME=[clfs]" > /etc/sysconfig/network
```

[clfs] needs to be replaced with the name given to the computer. Do not enter the Fully Qualified Domain Name (FQDN) here. That information will be put in the `/etc/hosts` file in the next section.

12.2. Customizing the /etc/hosts File

If a network card is to be configured, decide on the IP address, fully-qualified domain name (FQDN), and possible aliases for use in the `/etc/hosts` file. The syntax is:

```
<IP address> myhost.example.org aliases
```

Unless the computer is to be visible to the Internet (i.e., there is a registered domain and a valid block of assigned IP addresses—most users do not have this), make sure that the IP address is in the private network IP address range. Valid ranges are:

```
Private Network Address Range       Normal Prefix
10.0.0.1 - 10.255.255.254           8
172.x.0.1 - 172.x.255.254           16
192.168.y.1 - 192.168.y.254         24
```

x can be any number in the range 16-31. y can be any number in the range 0-255.

A valid IP address could be 192.168.1.1. A valid FQDN for this IP could be `www.cross-lfs.org` (not recommended because this is a valid registered domain address and could cause domain name server issues).

Even if not using a network card, a valid FQDN is still required. This is necessary for certain programs to operate correctly.

Create the `/etc/hosts` file by running:

```
cat > /etc/hosts << "EOF"
# Begin /etc/hosts (network card version)

127.0.0.1 localhost
[192.168.1.1] [<HOSTNAME>.example.org] [HOSTNAME] [alias ...]

# End /etc/hosts (network card version)
EOF
```

The *[192.168.1.1]* and *[<HOSTNAME>.example.org]* values need to be changed for specific users or requirements (if assigned an IP address by a network/system administrator and the machine will be connected to an existing network). The optional alias name(s) can be omitted.

If a network card is not going to be configured, create the `/etc/hosts` file by running:

```
cat > /etc/hosts << "EOF"
# Begin /etc/hosts (no network card version)

127.0.0.1 [<HOSTNAME>.example.org] [HOSTNAME] localhost

# End /etc/hosts (no network card version)
EOF
```

12.3. Creating the /etc/resolv.conf File

If the system is going to be connected to the Internet, it will need some means of Domain Name Service (DNS) name resolution to resolve Internet domain names to IP addresses, and vice versa. This is best achieved by placing the IP address of the DNS server, available from the ISP or network administrator, into `/etc/resolv.conf`. If at least one of your network interfaces is going to be configured by DHCP then you may not need to create this file. By default DHCPCD will overwrite this file when it gets a new lease from the DHCP server. If you wish to manually configure your network interfaces or manually set your DNS using DHCP then create the file by running the following:

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

domain [Your Domain Name]
nameserver [IP address of your primary nameserver]
nameserver [IP address of your secondary nameserver]

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

The *`domain`* statement can be omitted or replaced with a *`search`* statement. See the man page for resolv.conf for more details.

Replace *`[IP address of the nameserver]`* with the IP address of the DNS most appropriate for the setup. There will often be more than one entry (requirements demand secondary servers for fallback capability). If you only need or want one DNS server, remove the second *nameserver* line from the file. The IP address may also be a router on the local network.

12.4. DHCP or Static Networking?

This section only applies if a network card is to be configured. If you do not need to configure a network interface you can skip on to Making the CLFS System Bootable.

There are two different ways you can proceed from this point to configure your network. Dynamic will allow you to take advantage of a DHCP server to get all your configuration information. Static you become responsible for setting up your options.

To configure a Static Interface, Follow Section 12.5, “Static Networking Configuration”.

To configure a DHCP Interface, Follow Section 12.6, “DHCPCD-6.3.2”.

12.5. Static Networking Configuration

12.5.1. Creating the Static Network Interface Configuration Files

Which interfaces are brought up and down by the network script depends on the files and directories in the `/etc/sysconfig/network-devices` hierarchy. This directory should contain a sub-directory for each interface to be configured, such as `ifconfig.xyz`, where “xyz” is a network interface name. Inside this directory would be files defining the attributes to this interface, such as its IP address(es), subnet masks, and so forth.

The following command creates a sample `ipv4` file for the *eth0* device:

```
cd /etc/sysconfig/network-devices &&
mkdir -v ifconfig.eth0 &&
cat > ifconfig.eth0/ipv4 << "EOF"
ONBOOT="yes"
SERVICE="ipv4-static"
IP="192.168.1.1"
GATEWAY="192.168.1.2"
PREFIX="24"
BROADCAST="192.168.1.255"
EOF
```

The values of these variables must be changed in every file to match the proper setup. If the `ONBOOT` variable is set to “yes” the network script will bring up the Network Interface Card (NIC) during booting of the system. If set to anything but “yes” the NIC will be ignored by the network script and not be brought up.

The `SERVICE` variable defines the method used for obtaining the IP address. The CLFS-Bootscripts package has a modular IP assignment format, and creating additional files in the `/etc/sysconfig/network-devices/services` directory allows other IP assignment methods.

The `GATEWAY` variable should contain the default gateway IP address, if one is present. If not, then comment out the variable entirely.

The `PREFIX` variable needs to contain the number of bits used in the subnet. Each octet in an IP address is 8 bits. If the subnet's netmask is 255.255.255.0, then it is using the first three octets (24 bits) to specify the network number. If the netmask is 255.255.255.240, it would be using the first 28 bits. Prefixes longer than 24 bits are commonly used by DSL and cable-based Internet Service Providers (ISPs). In this example (PREFIX=24), the netmask is 255.255.255.0. Adjust the `PREFIX` variable according to your specific subnet.

To configure another DHCP Interface, Follow Section 12.6.2, “Creating the DHCP Network Interface Configuration Files”.

12.6. DHCPCD-6.3.2

The DHCPCD package provides a DHCP Client for network configuration.

12.6.1. Installation of DHCPCD

If you wish to configure your network to connect to a DHCP server, you will first need to install a DHCP client. CLFS uses the DHCPCD package for this.

Prepare DHCPCD for compilation:

```
./configure --prefix=/usr --sbindir=/sbin \
    --sysconfdir=/etc --dbdir=/var/lib/dhcpcd --libexecdir=/usr/lib/dhcpcd
```

Compile the package:

```
make
```

This package does not come with a test suite.

Install the package:

```
make install
```

12.6.2. Creating the DHCP Network Interface Configuration Files

First install the service from the CLFS Bootscripts package:

```
tar -xvf bootscripts-cross-lfs-3.0-20140710.tar.xz
cd bootscripts-cross-lfs-3.0-20140710
make install-service-dhcpcd
```

Finally, create the `/etc/sysconfig/network-devices/ifconfig.eth0/dhcpcd` configuration file using the following commands. Adjust appropriately for additional interfaces:

```
cd /etc/sysconfig/network-devices &&
mkdir -v ifconfig.eth0 &&
cat > ifconfig.eth0/dhcpcd << "EOF"
ONBOOT="yes"
SERVICE="dhcpcd"

# Start Command for DHCPCD
DHCP_START="-q"

# Stop Command for DHCPCD
DHCP_STOP="-k"
EOF
```

The values of these variables must be changed in every file to match the proper setup. If the `ONBOOT` variable is set to “yes” the network script will bring up the Network Interface Card (NIC) during booting of the system. If set to anything but “yes” the NIC will be ignored by the network script and not be brought up.

The `SERVICE` variable defines the method used for obtaining the IP address. The CLFS-Bootscripts package has a modular IP assignment format, and creating additional files in the `/etc/sysconfig/network-devices/services` directory allows other IP assignment methods.

The `DHCP_START` and `DHCP_STOP` variables arguments that are passed onto `dhcpcd` when starting and stoppping the service. More information about what can be passed can be found in the `dhcpcd(8)` man page.

To configure another Static Interface, Follow Section 12.5, “Static Networking Configuration”.

12.6.3. Contents of dhcpcd

Installed files: dhcpcd

Short Descriptions

dhcpcd dhcpcd is an implementation of the DHCP client specified in RFC 2131. It gets the host information from a DHCP server and configures the network interface automatically.

Chapter 13. Making the CLFS System Bootable

13.1. Introduction

It is time to make the CLFS system bootable. This chapter discusses building a kernel for the new CLFS system and installing the boot loader so that the CLFS system can be selected for booting at startup.

13.2. Linux-3.14.21

The Linux package contains the Linux kernel.

13.2.1. Installation of the kernel

Building the kernel involves a few steps—configuration, compilation, and installation. Read the `README` file in the kernel source tree for alternative methods to the way this book configures the kernel.

Apply the latest Linux sublevel patch:

```
xzcat ../patch-3.14.21.xz | patch -Np1 -i -
```

Prepare for compilation by running the following command:

```
make mrproper
```

This ensures that the kernel tree is absolutely clean. The kernel team recommends that this command be issued prior to each kernel compilation. Do not rely on the source tree being clean after un-tarring.

Note

A good starting place for setting up the kernel configuration is to run **make defconfig**. This will set the base configuration to a good state that takes your current system architecture into account.

Be sure to configure the following options as shown, or the system might not work correctly or boot at all:

```
Device Drivers  --->
  Generic Driver Options  --->
    ()  path to uevent helper (CONFIG_UEVENT_HELPER_PATH)
    [*] Maintain a devtmpfs filesystem to mount at /dev (CONFIG_DEVTMPFS)
    [ ] Fallback user-helper invocation for firmware loading (CONFIG_FW_LOA
```

Note

"EFI Variable support" and "EFI GUID Partition support" are for UEFI systems.

Configure the kernel via a menu-driven interface. CBLFS has some information regarding particular kernel configuration requirements of packages outside of CLFS at *http://cblfs.cross-lfs.org/*:

```
make menuconfig
```

Alternatively, **make oldconfig** may be more appropriate in some situations. See the `README` file for more information.

Warning

If you are using an existing config in which the ARCH was specified as `ppc` (instead of `powerpc`), you will have to run **make menuconfig** after **make oldconfig** and manually select many of the mac-specific options for ide and input.

If desired, skip kernel configuration by copying the kernel config file, `.config`, from the host system (assuming it is available) to the root directory of the unpacked kernel sources. However, we do not recommend this option. It is often better to explore all the configuration menus and create the kernel configuration from scratch.

Compile the kernel image and modules:

```
make
```

If using kernel modules, a configuration file in `/etc/modprobe.d` file may be needed. Information pertaining to modules and kernel configuration is located in the kernel documentation in the `Documentation` directory of the kernel sources tree. Also, `modprobe.d(5)` may be of interest.

Install the modules, if the kernel configuration uses them:

```
make modules_install
```

Install the firmware, if the kernel configuration uses them:

```
make firmware_install
```

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

Issue the following command to install the kernel:

```
cp -v vmlinux /boot/clfskernel-3.14.21
```

`System.map` is a symbol file for the kernel. It maps the function entry points of every function in the kernel API, as well as the addresses of the kernel data structures for the running kernel. Issue the following command to install the map file:

```
cp -v System.map /boot/System.map-3.14.21
```

The kernel configuration file `.config` produced by the **make menuconfig** step above contains all the configuration selections for the kernel that was just compiled. It is a good idea to keep this file for future reference:

```
cp -v .config /boot/config-3.14.21
```

It is important to note that the files in the kernel source directory are not owned by `root`. Whenever a package is unpacked as user `root` (like we do inside the final-system build environment), the files have the user and group IDs of whatever they were on the packager's computer. This is usually not a problem for any other package to be installed because the source tree is removed after the installation. However, the Linux source tree is often retained for a long time. Because of this, there is a chance that whatever user ID the packager used will be assigned to somebody on the machine. That person would then have write access to the kernel source.

If the kernel source tree is going to retained, run **chown -R 0:0** on the `linux-3.14` directory to ensure all files are owned by user `root`.

Warning

Some kernel documentation recommends creating a symlink from `/usr/src/linux` pointing to the kernel source directory. This is specific to kernels prior to the 2.6 series and *must not* be created on a CLFS system as it can cause problems for packages you may wish to build once your base CLFS system is complete.

Also, the headers in the system's `include` directory should *always* be the ones against which Glibc was compiled and should *never* be replaced by headers from a different kernel version.

13.2.2. Contents of Linux

Installed files:	config-[linux-version], clfskernel-[linux-version], and System.map-[linux-version]
Installed directory:	/lib/modules

Short Descriptions

`config-[linux-version]`	Contains all the configuration selections for the kernel
`clfskernel-[linux-version]`	The engine of the Linux system. When turning on the computer, the kernel is the first part of the operating system that gets loaded. It detects and initializes all components of the computer's hardware, then makes these components available as a tree of files to the software and turns a single CPU into a multitasking machine capable of running scores of programs seemingly at the same time.
`System.map-[linux-version]`	A list of addresses and symbols; it maps the entry points and addresses of all the functions and data structures in the kernel

13.3. Making the CLFS System Bootable

Your shiny new CLFS system is almost complete. One of the last things to do is to ensure that the system can be properly booted. The instructions below apply only to NewWorld Macintoshes.

Boot loading can be a complex area, so a few cautionary words are in order. Be familiar with the current boot loader and any other operating systems present on the hard drive(s) that need to be bootable. Make sure that an emergency CD is ready to “rescue” the computer if it becomes un-bootable. It is also a good idea to enable booting from Open Firmware in case things go really wrong.

Earlier, we compiled and installed the yaboot boot loader software in preparation for this step. The procedure involves writing the bootloader to a bootstrap partition and blessing it so that Open Firmware will boot from it. This is all handled by **ybin**, the yaboot installer.

Ybin writes an optional 'OS selector' menu into Open Firmware, then writes yaboot and `yaboot.conf` to the bootstrap partition, blesses this, and updates the boot device recorded in nvram. When booted, the OF provides the initial menu to choose between linux, boot from CD, and e.g. OSX (depending on what was in `yaboot.conf`). If you boot to 'linux', yaboot is executed and lets you select which kernel to use.

Images (kernels) are specified, together with any necessary path, in `yaboot.conf` - the details are incorporated into the bootloader, but no attempt is made to access or validate the paths until they are selected. There are many possible options that can be specified in `yaboot.conf`, see the man page for the details. Most people will be able to specify device=hd: (for a single hard disk), but if you have multiple disks, or if you wish to be pedantic, you can specify the full OF path to the device, obtained by running **ofpath /dev/hdX** .

Using the above information, determine the appropriate designators for the bootstrap partition and the root partition. For the following example, it is assumed that the bootstrap partition is `hda2` and the root partition is `hda7`. We will also assume that you wish to be able to boot an OSX installation on `hda4`. Change these items as necessary for your machine.

If your machine has a SATA disk, specify the partitions using `/dev/sda7` and so forth in the usual way. At least some of the distros specify a full OF path to the 'device' and to the image(s), such as *`device=/ht@0,f2000000/pci@3/k2-sata-root@c/k2-sata@0/disk@0:`* for the disk, and *`image=/ht@0,f2000000/pci@3/k2-sata-root@c/k2-sata@0/disk@0:9,/boot/clfskernel-3.14.21`* which definitely works.

Create a “yaboot.conf” file defining yaboot's boot menu:

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

# By default, yaboot will boot the first menu entry.

# Allow 10 seconds before booting the default.
# this will also apply to the first-stage os selector
timeout=100

# These variables are global
# first, where to put the bootstrap partition
boot=/dev/hda2

```

```
# Which disk to use
device=hd:

# Default partition for the kernel images
partition=7

# default root partition
root=/dev/hda7

# where ybin is to find yaboot and ofboot
install=/usr/lib/yaboot/yaboot
magicboot=/usr/lib/yaboot/ofboot

# allow the initial menu to offer CD as an option
enablecdboot

# allow the initial menu to offer booting from Open Firmware
enableofboot

# allow the initial menu to boot from mac osx
macosx=/dev/hda4

# white on black is boring!
# note  the spellings : 'fgcolor' but 'light'
# in this context, light means 'without high intensity'
fgcolor=light-green

# The first entry is for CLFS.
# For all images, the pathname is relative to the filesystem
# on which they are situated and can include at most one
# directory
image=/boot/clfskernel-3.14.21
    label=3.0.0-SYSVINIT
    read-only
EOF
```

Add an entry for the host distribution, if you have one. It might look something like this if the kernel and initrd are in the host's '/' directory on `hda6`:

```
cat >> /etc/yaboot.conf << "EOF"
title Debian
image=/pci@f4000000/ata-6d/disk@0:6,/vmlinux
    label=Debian
    initrd=/pci@f4000000/ata-6d/disk@0:6,/initrd.gz
    initrd-size=10000
    append="root=/dev/hda7"
    read-only
EOF
```

Warning

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

```
ybin
```

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

```
mkofboot
```

Chapter 14. The End

14.1. The End

Well done! The new CLFS system is installed! We wish you much success with your shiny new custom-built Linux system.

It may be a good idea to create an `/etc/clfs-release` file. By having this file, it is very easy for you (and for us if you need to ask for help at some point) to find out which CLFS version is installed on the system. Create this file by running:

```
echo 3.0.0-SYSVINIT > /etc/clfs-release
```

14.2. Download Client

The final system build does not install an FTP or HTTP client for downloading files.

Some suggested clients include:

- Curl *http://cblfs.cross-lfs.org/index.php/Curl*
- Inetutils *http://cblfs.cross-lfs.org/index.php/Inetutils*
- LFTP *http://lftp.yar.ru/*
- Links *http://cblfs.cross-lfs.org/index.php/Links*
- Lynx *http://cblfs.cross-lfs.org/index.php/Lynx*
- NcFTP Client *http://cblfs.cross-lfs.org/index.php/Ncftp*
- Wget *http://cblfs.cross-lfs.org/index.php/Wget*
- BASH - A user can use net redirections (if not disabled when building bash in the final system) to download wget or another program.

```
cat > download.sh << "EOF"
#!/bin/bash

WGET_VERSION='1.14'
WGET_HOSTNAME='ftp.gnu.org'
exec {HTTP_FD}<>/dev/tcp/${WGET_HOSTNAME}/80
echo -ne "GET /gnu/wget/wget-${WGET_VERSION}.tar.xz HTTP/1.1\r\nHost: "\
    ${WGET_HOSTNAME}'\r\nUser-Agent: '\
    'bash/'${BASH_VERSION}'\r\n\r\n'  >&${HTTP_FD}
sed -e '1,/^.$/d' <&${HTTP_FD} >wget-${WGET_VERSION}.tar.xz
EOF
```

- GAWK

```
cat > gawkdl.sh << "EOF"
#!/bin/bash

gawk 'BEGIN {
  NetService = "/inet/tcp/0/mirror.anl.gov/80"
  print "GET /pub/gnu/wget/wget-1.14.tar.xz" |& NetService
  while ((NetService |& getline) > 0)
    print $0
  close(NetService)
}' > binary

gawk '{q=p;p=$0}NR>1{print q}END{ORS = ""; print p}' binary > wget-1.14.tar.xz

rm binary
EOF
```

- PERL with HTTP::Tiny (Included with final system PERL install).

```
cat > download.pl << "EOF"
#!/usr/bin/perl

use HTTP::Tiny;
my $http = HTTP::Tiny->new;
my $response;

$response = $http->mirror('http://ftp.gnu.org/gnu/wget/wget-1.14.tar.xz', 'wget
die "Failed!\n" unless $response->{success};
print "Unchanged!\n" if $response->{status} eq '304';
EOF
```

  Or use this:

```
perl -MHTTP::Tiny -E 'say HTTP::Tiny->new->get(shift)->{content}' "http://ftp.g
perl -e 'local $/; $_ = <>; s/\n$//; print' binary > wget-1.14.tar.xz
rm binary
```

- PERL with LWP: Run **cpan** and manually configure the client. Run **install LWP** while in the CPAN shell.

  Refer to *http://www.bioinfo-user.org.uk/dokuwiki/doku.php/projects/wgetpl* for wgetpl.

14.3. Rebooting the System

If you built your final system using the boot method, just run **shutdown -r now** to reboot again, using your newly-built kernel instead of the miminal one currently in use. If you chrooted, there are a few more steps.

The system you have created in this book is quite minimal, and most likely will not have the functionality you would need to be able to continue forward. By installing a few extra packages from CBLFS while still in our current chroot environment, you can leave yourself in a much better position to continue on once you reboot into your new CLFS installation. Installing a text mode web browser, such as Lynx, you can easily view the CBLFS website in one virtual terminal, while building packages in another. The GPM package will also allow you to perform copy/paste actions in your virtual terminals. Lastly, if you are in a situation where static IP configuration does not meet your networking requirements, installing packages such as 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

if [ -h ${CLFS}/dev/shm ]; then
  link=$(readlink ${CLFS}/dev/shm)
  umount -v ${CLFS}/$link
  unset link
else
  umount -v ${CLFS}/dev/shm
fi

umount ${CLFS}/dev
umount ${CLFS}/proc
umount ${CLFS}/sys
umount ${CLFS}/run
```

Unmount the CLFS file system itself:

```
umount ${CLFS}
```

If multiple partitions were created, unmount the other partitions before unmounting the main one, like this:

```
umount ${CLFS}/usr
umount ${CLFS}/home
umount ${CLFS}
```

Now, reboot the system with:

```
shutdown -r now
```

Assuming the boot loader was set up as outlined earlier, *CLFS 3.0.0-SYSVINIT* will boot automatically.

When the reboot is complete, the CLFS system is ready for use and more software may be added to suit your needs.

14.4. What Now?

Thank you for reading this CLFS book. We hope that you have found this book helpful and have learned more about the system creation process.

Now that the CLFS system is installed, you may be wondering “What next?” To answer that question, we have compiled a list of resources for you.

- Maintenance

Bugs and security notices are reported regularly for all software. Since a CLFS system is compiled from source, it is up to you to keep abreast of such reports. There are several online resources that track such reports, some of which are shown below:

- Freecode (*http://freecode.com/*)

  Freecode can notify you (via email) of new versions of packages installed on your system.

- *CERT* (Computer Emergency Response Team)

  CERT has a mailing list that publishes security alerts concerning various operating systems and applications. Subscription information is available at *http://www.us-cert.gov/cas/signup.html*.

- Bugtraq

  Bugtraq is a full-disclosure computer security mailing list. It publishes newly discovered security issues, and occasionally potential fixes for them. Subscription information is available at *http://www.securityfocus.com/archive*.

- Community Driven Beyond Linux From Scratch

  The Community Driven Beyond Linux From Scratch wiki covers installation procedures for a wide range of software beyond the scope of the CLFS Book. CBLFS is designed specifically to work with the CLFS book, and has all the necessary information to continue the builds in the same manner that CLFS uses. This is a community driven project, which means anyone can contribute and provide updates. The CBLFS project is located at *http://cblfs.cross-lfs.org/*.

- CLFS Hints

  The CLFS Hints are a collection of educational documents submitted by volunteers in the CLFS community. The hints are available at *http://hints.cross-lfs.org/index.php/*.

- Mailing lists

  There are several CLFS mailing lists you may subscribe to if you are in need of help, want to stay current with the latest developments, want to contribute to the project, and more. See Chapter 1 - Mailing Lists for more information.

- The Linux Documentation Project

  The goal of The Linux Documentation Project (TLDP) is to collaborate on all of the issues of Linux documentation. The TLDP features a large collection of HOWTOs, guides, and man pages. It is located at *http://www.tldp.org/*.

Part VI. Appendices

Appendix A. Acronyms and Terms

ABI	Application Binary Interface
API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
ATA	Advanced Technology Attachment (see IDE)
BIOS	Basic Input/Output System
bless	manipulate a filesystem so that OF will boot from it
BSD	Berkeley Software Distribution
CBLFS	Community Driven Beyond Linux From Scratch
chroot	change root
CLFS	Cross-Compiled Linux From Scratch
CMOS	Complementary Metal Oxide Semiconductor
COS	Class Of Service
CPU	Central Processing Unit
CRC	Cyclic Redundancy Check
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name Service
EGA	Enhanced Graphics Adapter
ELF	Executable and Linkable Format
EOF	End of File
EQN	equation
ext2	second extended file system
ext3	third extended file system
ext4	fourth extended file system
FAQ	Frequently Asked Questions
FHS	Filesystem Hierarchy Standard
FIFO	First-In, First Out
FQDN	Fully Qualified Domain Name
FTP	File Transfer Protocol
GB	Gigabytes
GCC	GNU Compiler Collection
GID	Group Identifier
GMT	Greenwich Mean Time
HTML	Hypertext Markup Language
IDE	Integrated Drive Electronics

IEEE	Institute of Electrical and Electronic Engineers
IO	Input/Output
IP	Internet Protocol
IPC	Inter-Process Communication
IRC	Internet Relay Chat
ISO	International Organization for Standardization
ISP	Internet Service Provider
KB	Kilobytes
LED	Light Emitting Diode
LFS	Linux From Scratch
LSB	Linux Standard Base
MB	Megabytes
MBR	Master Boot Record
MD5	Message Digest 5
NIC	Network Interface Card
NLS	Native Language Support
NPTL	Native POSIX Threading Library
OF	Open Firmware
OSS	Open Sound System
PCH	Pre-Compiled Headers
PID	Process Identifier
PTY	pseudo terminal
QA	Quality Assurance
QOS	Quality Of Service
RAM	Random Access Memory
RPC	Remote Procedure Call
RTC	Real Time Clock
SCO	The Santa Cruz Operation
SATA	Serial ATA
SGR	Select Graphic Rendition
SHA1	Secure-Hash Algorithm 1
TLDP	The Linux Documentation Project
TFTP	Trivial File Transfer Protocol
TLS	Thread-Local Storage
UID	User Identifier
umask	user file-creation mask

USB Universal Serial Bus

UTC Coordinated Universal Time

UUID Universally Unique Identifier

VC Virtual Console

VGA Video Graphics Array

VT Virtual Terminal

Appendix B. Dependencies

Every package built in CLFS relies on one or more other packages in order to build and install properly. Some packages even participate in circular dependencies, that is, the first package depends on the second which in turn depends on the first. Because of these dependencies, the order in which packages are built in CLFS is very important. The purpose of this page is to document the dependencies of each package built in CLFS.

For each package we build, we have listed three types of dependencies. The first lists what other packages need to be available in order to compile and install the package in question. The second lists what packages, in addition to those on the first list, need to be available in order to run the test suites. The last list of dependencies are packages that require this package to be built and installed in its final location before they are built and installed. In most cases, this is because these packages will hardcode paths to binaries within their scripts. If not built in a certain order, this could result in paths of /tools/bin/[binary] being placed inside scripts installed to the final system. This is obviously not desirable.

Autoconf

Installation depends on: Bash, Coreutils, Gawk, Grep, M4, Make, Perl, Sed, Texinfo

Test suite depends on: Automake, Binutils, Diffutils, Findutils, GCC, Libtool

Must be installed before: Automake

Automake

Installation depends on: Autoconf, Bash, Binutils, Coreutils, Gawk, Grep, M4, Make, Perl, Sed, Texinfo

Test suite depends on: Bison, Bzip2, DejaGNU, Diffutils, Expect, Findutils, Flex, GCC, Gettext, Gzip, Libtool, Tar, XZ Utils. Can also use several other packages that are not installed in CLFS.

Must be installed before: None

Bash

Installation depends on: Bash, Bison, Coreutils, Diffutils, Glibc, Gawk, GCC, Grep, Make, Ncurses, Patch, Readline, Sed, Texinfo

Test suite depends on: None

Must be installed before: None

Bc

Installation depends on: Bash, Binutils, Bison, Coreutils, Glibc, GCC, Grep, Make, Readline

Test suite depends on: Gawk

Must be installed before: None

Binutils

Installation depends on: Bash, Binutils, Coreutils, Diffutils, Glibc, File, Gawk, GCC, Grep, Make, Perl, Sed, Texinfo, Zlib

Test suite depends on: DejaGNU, Expect

Must be installed before: None

Bison

Installation depends on: Bash, Binutils, Coreutils, Glibc, Gawk, GCC, Grep, M4, Make, Sed

Test suite depends on: Diffutils, Findutils, Gawk

Must be installed before: Flex, Kbd, Tar

Bzip2

Installation depends on: Bash, Binutils, Coreutils, Glibc, GCC, Make

Test suite depends on: Diffutils

Must be installed before: None

CLFS-Bootscripts

Installation depends on: Bash, Coreutils, Make, Sed

Test suite depends on: None

Must be installed before: None

Check

Installation depends on: GCC, Grep, Make, Sed, Texinfo

Test suite depends on: None

Must be installed before: None

CLooG

Installation depends on: Bash, Binutils, Coreutils, Diffutils, Glibc, Gawk, GCC, Grep, GMP, ISL, Make, MPC, MPFR, Sed, Texinfo

Test suite depends on: None

Must be installed before: GCC

Coreutils

Installation depends on: Bash, Binutils, Coreutils, Glibc, Gawk, GCC, GMP, Grep, Make, Patch, Perl, Sed, Texinfo

Test suite depends on: Diffutils, E2fsprogs, Findutils, Util-linux

Must be installed before: Bash, Diffutils, Eudev, Findutils, Man

DejaGNU

Installation depends on: Bash, Coreutils, Diffutils, GCC, Grep, Make, Sed

Test suite depends on: None

Must be installed before: None

DHCPCD

Installation depends on: Bash, Coreutils, GCC, Make, Sed

Test suite depends on: No test suite available

Must be installed before: None

Diffutils

Installation depends on: Bash, Binutils, Coreutils, Glibc, GCC, Grep, Make, Patch, Sed, Texinfo

Test suite depends on: No test suite available

Must be installed before: None

Eudev

Installation depends on: Binutils, Coreutils, Diffutils, Glibc, Gawk, GCC, Grep, Make, Sed

Test suite depends on: No test suite available

Must be installed before: Systemd

Expect

Installation depends on: Bash, Binutils, Coreutils, Diffutils, Glibc, GCC, Grep, Make, Patch, Sed, Tcl

Test suite depends on: None

Must be installed before: None

E2fsprogs

Installation depends on: Bash, Binutils, Coreutils, Glibc, Gawk, GCC, Gettext, Grep, Gzip, Make, Pkg-config-lite, Sed, Texinfo, Util-linux

Test suite depends on: Bzip2, Diffutils

Must be installed before: None

File

Installation depends on: Bash, Binutils, Coreutils, Diffutils, Glibc, Gawk, GCC, Grep, Make, Sed, Zlib

Test suite depends on: No test suite available

Must be installed before: None

Findutils

Installation depends on: Bash, Binutils, Coreutils, Glibc, GCC, Grep, Make, Sed, Texinfo

Test suite depends on: DejaGNU, Diffutils, Expect

Must be installed before: None

Flex

Installation depends on: Bash, Binutils, Coreutils, Glibc, GCC, Grep, M4, Make, Sed, Texinfo

Test suite depends on: Bison, Diffutils, Gawk

Must be installed before: IPRoute2, Kbd, Man

Gawk

Installation depends on: Bash, Binutils, Coreutils, Glibc, GCC, Grep, Make, Sed, Texinfo

Test suite depends on: Diffutils

Must be installed before: None

Gcc

Installation depends on: Bash, Binutils, CLooG, Coreutils, Diffutils, Glibc, Findutils, Gawk, GCC, GMP, Grep, ISL, Make, MPFR, Patch, Perl, Sed, Tar, Texinfo

Test suite depends on: Check, DejaGNU, Expect

Must be installed before: None

GDBM

Installation depends on: Bash, Binutils, Coreutils, Diffutils, GCC, Grep, Make, Sed

Test suite depends on: None

Must be installed before: None

Gettext

Installation depends on: Bash, Binutils, Coreutils, Diffutils, Glibc, Findutils, Gawk, GCC, Grep, Make, Sed, Texinfo

Test suite depends on: Tar, Tcl

Must be installed before: Automake

Glibc

Installation depends on: Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Gettext, Grep, Gzip, Make, Perl, Sed, Texinfo

Test suite depends on: None

Must be installed before: None

GMP

Installation depends on: Bash, Binutils, Coreutils, Diffutils, Glibc, Gawk, GCC, Grep, M4, Make, Sed, Texinfo

Test suite depends on: None

Must be installed before: MPFR, GCC

Grep

Installation depends on: Bash, Binutils, Coreutils, Glibc, GCC, Grep, Make, Patch, Sed, Texinfo

Test suite depends on: Diffutils, Gawk

Must be installed before: Man

Groff

Installation depends on: Bash, Binutils, Coreutils, Glibc, Gawk, GCC, Grep, Make, Perl Sed, Texinfo

Test suite depends on: No test suite available

Must be installed before: Man, Perl

Gzip

Installation depends on: Bash, Binutils, Coreutils, Glibc, GCC, Grep, Make, Sed, Texinfo

Test suite depends on: Diffutils

Must be installed before: Man

Iana-Etc

Installation depends on: Coreutils, Gawk, Make

Test suite depends on: No test suite available

Must be installed before: Perl

IProute2

Installation depends on: Bash, Binutils, Bison, Coreutils, Glibc, Findutils, Flex, GCC, Make, Linux Headers, Sed

Test suite depends on: No test suite available

Must be installed before: None

IPutils

Installation depends on: Bash, Binutils, Coreutils, Glibc, GCC, Make

Test suite depends on: No test suite available

Must be installed before: None

ISL

Installation depends on: Bash, Binutils, Coreutils, Diffutils, Glibc, Gawk, GCC, Grep, GMP, Make, MPC, MPFR, Sed, Texinfo

Test suite depends on: None

Must be installed before: GCC

Kbd

Installation depends on: Bash, Binutils, Check, Coreutils, Glibc, Gawk, GCC, Gzip, Make

Test suite depends on: No test suite available

Must be installed before: None

KMOD

Installation depends on: Bash, Binutils, Bison, Coreutils, Glibc, Flex, Gawk, GCC, Gettext, Gzip, Make, Pkg-config-lite, Sed, XZ Utils, Zlib.

Test suite depends on: No test suite available

Must be installed before: Eudev

Less

Installation depends on: Bash, Binutils, Coreutils, Glibc, GCC, Grep, Make, Ncurses, Sed

Test suite depends on: No test suite available

Must be installed before: None

Libee

Installation depends on: Bash, Binutils, Coreutils, Diffutils, Glibc, Findutils, Gawk, GCC, Grep, Libestr, Make, Pkg-config-lite, Sed, Texinfo

Test suite depends on: No test suite available

Must be installed before: Rsyslog

Libestr

Installation depends on: Bash, Binutils, Coreutils, Diffutils, Glibc, Findutils, Gawk, GCC, Grep, Make, Sed, Texinfo

Test suite depends on: None

Must be installed before: Libee, Rsyslog

Libpipeline

Installation depends on: Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Make, Sed, Texinfo

Test suite depends on: Check

Must be installed before: Man-DB

Libtool

Installation depends on: Bash, Binutils, Coreutils, Diffutils, Glibc, Findutils, Gawk, GCC, Grep, Make, Sed, Texinfo

Test suite depends on: Autoconf

Must be installed before: None

Linux Headers

Installation depends on: Binutils, Coreutils, Findutils, GCC, Grep, Make, Perl, Sed

Test suite depends on: No test suite available

Must be installed before: None

Linux Kernel

Installation depends on: Bash, Binutils, Coreutils, Diffutils, Glibc, Findutils, GCC, Grep, Gzip, Make, KMOD, Ncurses, Perl, Sed

Test suite depends on: No test suite available

Must be installed before: None

M4

Installation depends on: Bash, Binutils, Coreutils, Glibc, Gawk, GCC, Grep, Make, Sed, Texinfo

Test suite depends on: Diffutils

Must be installed before: Autoconf, Bison

Make

Installation depends on: Bash, Binutils, Coreutils, Glibc, GCC, Grep, Make, Sed, Texinfo

Test suite depends on: Perl, Procps

Must be installed before: None

Man-DB

Installation depends on: Bash, Binutils, Bzip2, Coreutils, Glibc, Gawk, GCC, Grep, Groff, Gzip, Less, XZ Utils, Make, Sed

Test suite depends on: No test suite available

Must be installed before: None

Man-Pages

Installation depends on: Bash, Coreutils, Make

Test suite depends on: No test suite available

Must be installed before: None

MPC

Installation depends on: Bash, Binutils, Coreutils, Diffutils, Glibc, Gawk, GCC, Grep, GMP, Make, MPFR, Sed, Texinfo

Test suite depends on: None

Must be installed before: GCC

MPFR

Installation depends on: Bash, Binutils, Coreutils, Diffutils, Glibc, Gawk, GCC, Grep, GMP, Make, Sed, Texinfo

Test suite depends on: None

Must be installed before: GCC

KMOD

Installation depends on: Bash, Binutils, Coreutils, Glibc, Findutils, GCC, Grep, Make, Sed, Zlib

Test suite depends on: Diffutils, File, Gawk, Gzip

Must be installed before: None

Ncurses

Installation depends on:	Bash, Binutils, Coreutils, Diffutils, Glibc, Gawk, GCC, Grep, Make, Pkg-config-lite, Sed
Test suite depends on:	No test suite available
Must be installed before:	Bash, GRUB, Inetutils, Less, Procps, Psmisc, Readline, Texinfo, Util-linux, Vim

Patch

Installation depends on:	Bash, Binutils, Coreutils, Glibc, GCC, Grep, Make, Sed
Test suite depends on:	No test suite available
Must be installed before:	None

Perl

Installation depends on:	Bash, Binutils, Bzip2, Coreutils, Glibc, Gawk, GCC, Grep, Make, Sed
Test suite depends on:	Gzip, Iana-Etc, Procps, Tar
Must be installed before:	Autoconf

Pkg-config-lite

Installation depends on:	Bash, Binutils, Coreutils, Diffutils, Glibc, Gawk, GCC, Grep, Make, Sed
Test suite depends on:	None
Must be installed before:	Util-linux, E2fsprogs

Procps

Installation depends on:	Bash, Binutils, Coreutils, Glibc, GCC, Make, Ncurses
Test suite depends on:	No test suite available
Must be installed before:	None

Psmisc

Installation depends on:	Bash, Binutils, Coreutils, Glibc, GCC, Grep, Make, Ncurses, Sed
Test suite depends on:	No test suite available
Must be installed before:	None

Readline

Installation depends on:	Bash, Binutils, Coreutils, Glibc, GCC, Grep, Make, Ncurses, Patch, Sed, Texinfo
Test suite depends on:	No test suite available
Must be installed before:	Bash

Rsyslog

Installation depends on:	Binutils, Coreutils, Diffutils, Glibc, Gawk, GCC, Grep, libee, Libestr, Make, Sed, Zlib

Test suite depends on: No testsuite available

Must be installed before: None

Sed

Installation depends on: Bash, Binutils, Coreutils, Glibc, GCC, Grep, Make, Sed, Texinfo

Test suite depends on: Diffutils, Gawk

Must be installed before: E2fsprogs, File, Libtool, Shadow

Shadow

Installation depends on: Bash, Binutils, Coreutils, Diffutils, Glibc, Findutils, Gawk, GCC, Gettext, Grep, Make, Sed

Test suite depends on: No test suite available

Must be installed before: None

Sysvinit

Installation depends on: Binutils, Coreutils, Glibc, GCC, Make, Sed

Test suite depends on: No test suite available

Must be installed before: None

Tar

Installation depends on: Bash, Binutils, Bison, Coreutils, Glibc, GCC, Grep, Make, Sed, Texinfo

Test suite depends on: Diffutils, Findutils, Gawk, Gzip

Must be installed before: None

Tcl

Installation depends on: Bash, Binutils, Coreutils, Diffutils, Glibc, GCC, Grep, Make, Sed

Test suite depends on: None

Must be installed before: None

Texinfo

Installation depends on: Bash, Binutils, Coreutils, Glibc, Gawk, GCC, Grep, Make, Ncurses, Sed

Test suite depends on: Diffutils, Gzip

Must be installed before: None

Eudev

Installation depends on: Binutils, Coreutils, Diffutils, Glibc, Gawk, GCC, Grep, Make, Sed

Test suite depends on: No test suite available

Must be installed before: None

Util-linux

Installation depends on: Bash, Binutils, Coreutils, Glibc, GCC, Grep, Make, Ncurses, Pkg-config-lite, Sed, Texinfo, Zlib

Test suite depends on: No testsuite available

Must be installed before: E2fsprogs

Vim

Installation depends on: Bash, Binutils, Coreutils, Diffutils, Glibc, Findutils, Gawk, GCC, Gettext, Grep, Make, Ncurses, Perl, Sed

Test suite depends on: Gzip

Must be installed before: None

XZ Utils

Installation depends on: Bash, Binutils, Coreutils, Diffutils, Glibc, Findutils, Gawk, GCC, Grep, Make, Sed

Test suite depends on: None

Must be installed before: None

Zlib

Installation depends on: Bash, Binutils, Coreutils, Glibc, GCC, Make, Sed

Test suite depends on: None

Must be installed before: File, KMOD, Util-linux

Appendix C. PowerPC Dependencies

This page contains dependency information for packages specific to ppc.

hfsutils

Installation depends on: Bash, Binutils, Coreutils, GCC, Make

Test suite depends on: None

Must be installed before: None

Parted

Installation depends on: Bash, Binutils, Coreutils, E2fsprogs, GCC, Make, Ncurses, Readline

Test suite depends on: None

Must be installed before: None

Powerpc-Utils

Installation depends on: Binutils, GCC, Make, Patch

Test suite depends on: None

Must be installed before: None

Yaboot

Installation depends on: Binutils, Coreutils, GCC, Make, Mktemp, Patch, Sed

Test suite depends on: None

Must be installed before: None

Appendix D. Package Rationale

CLFS includes many packages, a number of which might not necessarily be required for a "minimal" system, but still considered very useful. The purpose of this page is to list the reasoning for each package's inclusion in the book.

- Autoconf

  The Autoconf package contains programs for producing shell scripts that can automatically configure source code. This is useful for software developers, as well as anyone who wants to install packages that don't come with a configure script, such as some of the packages in CBLFS.
- Automake

  The Automake package contains programs for generating Makefiles for use with Autoconf. This can be useful to software developers.
- Bash

  This package contains the Bourne-Again SHell. A shell is an important component of a Linux system, as there must be some way of allowing the users to enter commands.
- Bc

  This package contains a precision calculator. The Linux kernel uses Bc to render the timeconst header.
- Binutils

  This package contains programs for handling object files. The programs in this package are needed for compiling most of the packages in CLFS.
- Bison

  This package contains programs that are required by several packages in CLFS.
- Bzip2

  The programs in this package are useful for compressing files to reduce size. They are also needed to uncompress tarballs for many CLFS packages.
- CLFS-Bootscripts

  This package contains a number of scripts that run at boottime, performing essential tasks such as mounting/ checking filesystems and starting the network interface.
- Check

  This package contains a test harness for other programs. It is used for some packages' test suites.
- CLooG

  This package is used by GCC to enable its Graphite loop generation code.
- Coreutils

  This package contains many basic command-line file-management tools, required for installation of every package in CLFS.
- DejaGNU

  This package is needed for the test suites of several packages, especially GCC and Binutils.
- DHCPCD

This package allows for automatic configuration of network interfaces from a DHCP server. It (or some other package providing a DHCP client is needed to connect to a DHCP server.

- Diffutils

  This package contains programs to compare files, and can also be used to create patches. It is required by the installation procedures of many CLFS packages, and used by many packages' test suites.

- Eudev

  This is a package that allows for dynamic creation of device nodes. It is a fork of Udev, which is now part of Systemd. It is still used for the "Boot" method in the temp-system, as Systemd is not needed there.

- Expect

  This package is needed for the test suites for several packages.

- E2fsprogs

  The programs in this package are used for the creation and maintenance of ext2/3/4 filesystems.

- File

  This package contains a program that determines the type of a given file. It is needed by some CLFS packages.

- Findutils

  This package contains programs for finding files based on certain criteria, and optionally performing commands on them. These programs are used by the installation procedures of many CLFS packages.

- Flex

  This package contains a tool for generating text scanners. It is used by multiple packages in CLFS

- Gawk

  This package contains programs for manipulating text files, using the AWK language. It is used by the installation procedures of many packages in CLFS.

- Gcc

  This package contains a C compiler, which is required to compile most of the packages in CLFS.

- GDBM

  This package contains the GNU Database Manager library. Man-DB requires either GDBM or Berkeley DB, though it prefers GDBM.

- Gettext

  A tool that allows programmers to easily implement i18n (internationalization) in their programs. It is a required dependency for a number of packages

- Glibc

  Any dynamically-linked C program (nearly every package in CLFS has these) needs a C library to compile and run.

- GMP

  This package is required by GCC.

- Grep

This package contains programs for searching for text in files. These programs are required by many packages in CLFS.

- Groff

  This package is required by Man-DB.

- Gzip

  Useful for compressing files to reduce size. It is also needed to uncompress tarballs for many CLFS packages

- Iana-Etc

  This package provides the `/etc/services` and `/etc/protocols` files. These files map port names to port numbers as well as protocol names to their corresponding numbers. These files are essential for many network based programs to work properly.

- IProute2

  This package contains programs for administering network interfaces.

- IPutils

  This package contains several basic network-management tools.

- ISL

  This package is required by CLooG.

- Kbd

  Contains keytable files and keyboard utilities compatible with the Linux kernel. These can be used to change the display font and keyboard layout.

- Kmod

  This package contains programs that assist in loading an unloading kernel modules.

- Less

  A program that lets you view text files one page at a time. It is also used by Man-DB for displaying manpages.

- Libee

  This library is required by Rsyslog.

- Libestr

  This library is required by Rsyslog.

- Libpipeline

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

- Libtool

  The Libtool package contains the GNU generic library support script. It is used by some CLFS packages.

- Linux Headers

  This package consists of santized headers from the Linux Kernel. These headers are required for Glibc to compile.

- Linux Kernel

  The Linux operating system.
- M4

  This package contains a macro processor. It is required by several CLFS packages, including Bison.
- Make

  This is required for installation of most CLFS packages
- Man-DB

  This package contains programs for finding and viewing man pages, and has superior internationalization capabilities compared to the Man package.
- Man-Pages

  A number of useful manpages, not supplied by other packages
- MPC

  This package is required by GCC.
- MPFR

  This package is required by GCC.
- Ncurses

  Needed by several packages in CLFS, such as Vim, Bash, and Less
- Patch

  Used for applying patches in several CLFS packages
- Perl

  The Perl package contains the Practical Extraction and Report Language. It is required by several CLFS packages.
- Pkg-config-lite

  Several packages in CLFS, and many others outside of CLFS, use **pkg-config** to locate dependencies.
- Procps-ng

  Provides a number of small, useful utilities that give information about the `/proc` filesystem.
- Psmisc

  Provides more utilties that give information about the `/proc` filesystem.
- Readline

  The Readline library provides a set of functions for use by applications that allow users to edit command lines as they are typed in. This is essential for input in programs like **bash** to work properly.
- Rsyslog

  This package provides a system logging daemon.
- Sed

  This package contains a stream editor. It is used in the installation procedures of most CLFS packages.

- Shadow

  This package contains programs that assist in the administration of users and groups, and passwords.
- Sysvinit

  Sysvinit provides the init daemon for the system.
- Tar

  Required to unpack the tar archives in which all CLFS packages are distributed
- Tcl

  Needed for the test suites of several packages
- Texinfo

  This package contains programs for viewing, installing and converting info pages. It is used in the installation procedures of many CLFS packages.
- Util-linux

  The Util-linux package contains miscellaneous utility programs. Among them are utilities for handling file systems, consoles, partitions, and messages. It also includes libraries that are required by E2fsprogs.
- Vim

  The Vim package contains a text editor. Users may substitute Nano, Joe, Emacs, or whatever other editor they prefer.
- XZ Utils

  Useful for compressing files to reduce size. Also needed to uncompress tarballs for many CLFS packages
- Zlib

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

Appendix E. Open Firmware and Mac issues.

This appendix documents some of the features of ppc macintoshes, and in particular the requirements of coexisting with Mac OS's (OSX or the old OS9). It is only relevant to NewWorld hardware.

Open Firmware and blessed partitions

The Open Firmware (OF) is the code in ROM or nvram which controls how the machine boots. If booting automatically, it will boot from the first valid blessed partition it finds (this is a simplification, but it is adequate for normal purposes).

It can only read apple filesystems (hfs, hfs+, or hfsx depending on the version of the firmware). For disks under linux, the blessing is done by ybin when it installs yaboot (the loader) and yaboot.conf.

Mac OS's have a tendency to look at other hfs{,+,x} filesystems on the disk, and unbless them if they do not match their expectations. Unblessing makes them unbootable. Fortunately, a filesystem of type `Apple_Bootstrap` can be read as hfs by the OF, but will be ignored by Mac OS.

Partitioning

Macintoshes use their own partition format - this means that other machines are unlikely to be able to read or write to macintosh partitions (in particular, fdisk does not understand them). The format allows a large number of individual partitions, and the native Mac tools had a tendency to insert small "filler" partitions between the real partitions. Under linux, using more than 15 partitions can be problematic (shortage of device nodes), so the normal approach is to use the Mac tools to create an area of freespace at the *front* of the disk, then put the Mac OS partition(s) after it and (re-)install the Mac OS. The freespace can then be partitioned using **parted** or the older **mac-fdisk**. It seems that recent versions of the Mac tools may no longer insert the filler partitions, so it may be possible to do all the partitioning before installing OSX.

> **Warning**
>
> The Macintosh resizing and partitioning tools are destructive and may delete all data when a partition is resized, even on unaltered partitions.

For the Linux partitions, you will need a bootstrap partition - this can normally be a mere 800KB in size (the smallest hfs partition available) although the Fedora installer has been known to insist on 800MB. This has to be in front of the Mac OS partition. The bootstrap is *never* mounted as a regular partition and should not be confused with a `/boot` partition. Other partitions are as normal (at least one rootfs, perhaps swap, perhaps others).

According to the lfs-from-osx hint, the Mac partitioning tools can create an apple_bootstrap partition and therefore there is no need to use a Linux CD to create the desired partitions from freespace, but using a Linux CD to create the partitions is a more widely tested approach.

If you follow this approach, partition 1 will be the apple partition map, partition 2 will be the bootstrap at the start of the disk, the linux partitions will follow, and then the mac partition(s) - under OSX the first mac partition will be number 3, under OS9 it would have a higher number and there would be some apple driver partitions.

OSX or OF upgrades

If the machine is dual-booted with OSX, the mac kernel or the OF will probably be upgraded at some point. This appears to either unbless the bootstrap, or else just point the OF boot device to the mac partition - so, the linux system will no longer be bootable.

Therefore, you will need to know which partition contains the bootstrap so that you can boot it from OF (on an apple keyboard, hold down option-command-o-f (that is, alt-apple-o-f) while booting then enter a command like:

```
boot hd:2,yaboot
```

This will allow you to select a linux boot, and from there you will have to rerun **ybin**.

The "OS chooser" menu that yaboot typically loads is stored in the OF and will not be available after a Mac kernel or firmware upgrade until **ybin** has been rerun.

Yaboot's requirements

Yaboot is the boot loader for linux, sometimes referred to as the second stage loader. It reads the yaboot.conf file on the bootstrap partition to find which linux system(s) should be available, and attempts to load the required kernel.

The bootstrap man page warns that the path to the kernel should contain no more than one directory for reliability.

Yaboot has to be able to understand the filesystem, so that it can find the kernel. It understands hfs (not useful for linux, it is not case-sensitive), ext2 (and therefore it can read ext3), reiser3, and xfs. If you want to use a different type of filesystem for '/' you will have to create a separate boot partition with a supported filesystem, and use that to hold the kernels.

Requirements if starting from OSX

Older versions of OSX (panther, leopard) can write to ext2 filesystems using version 1.3 of ext2fsx. The upgrade to tiger broke this, and version 1.4 of ext2fsx only supports reading. Users of current OSX will therefore have to find some other way of creating a suitable filesystem and populating it, such as a Live CD or rescue CD.

Appendix F. Open Publication License

v1.0, 8 June 1999

I. REQUIREMENTS ON BOTH UNMODIFIED AND MODIFIED VERSIONS

The Open Publication works may be reproduced and distributed in whole or in part, in any medium physical or electronic, provided that the terms of this license are adhered to, and that this license or an incorporation of it by reference (with any options elected by the author(s) and/or publisher) is displayed in the reproduction.

Proper form for an incorporation by reference is as follows:

> Copyright © <year> by <author's name or designee>. This material may be distributed only subject to the terms and conditions set forth in the Open Publication License, vX.Y or later (the latest version is presently available at *http://www.opencontent.org/openpub/*).

The reference must be immediately followed with any options elected by the author(s) and/or publisher of the document (see section VI).

Commercial redistribution of Open Publication-licensed material is permitted.

Any publication in standard (paper) book form shall require the citation of the original publisher and author. The publisher and author's names shall appear on all outer surfaces of the book. On all outer surfaces of the book the original publisher's name shall be as large as the bridgehead of the work and cited as possessive with respect to the bridgehead.

II. COPYRIGHT

The copyright to each Open Publication is owned by its author(s) or designee.

III. SCOPE OF LICENSE

The following license terms apply to all Open Publication works, unless otherwise explicitly stated in the document.

Mere aggregation of Open Publication works or a portion of an Open Publication work with other works or programs on the same media shall not cause this license to apply to those other works. The aggregate work shall contain a notice specifying the inclusion of the Open Publication material and appropriate copyright notice.

SEVERABILITY. If any part of this license is found to be unenforceable in any jurisdiction, the remaining portions of the license remain in force.

NO WARRANTY. Open Publication works are licensed and provided "as is" without warranty of any kind, express or implied, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose or a warranty of non-infringement.

IV. REQUIREMENTS ON MODIFIED WORKS

All modified versions of documents covered by this license, including translations, anthologies, compilations and partial documents, must meet the following requirements:

1. The modified version must be labeled as such.
2. The person making the modifications must be identified and the modifications dated.

3. Acknowledgement of the original author and publisher if applicable must be retained according to normal academic citation practices.
4. The location of the original unmodified document must be identified.
5. The original author's (or authors') name(s) may not be used to assert or imply endorsement of the resulting document without the original author's (or authors') permission.

V. GOOD-PRACTICE RECOMMENDATIONS

In addition to the requirements of this license, it is requested from and strongly recommended of redistributors that:

1. If you are distributing Open Publication works on hardcopy or CD-ROM, you provide email notification to the authors of your intent to redistribute at least thirty days before your manuscript or media freeze, to give the authors time to provide updated documents. This notification should describe modifications, if any, made to the document.
2. All substantive modifications (including deletions) be either clearly marked up in the document or else described in an attachment to the document.
3. Finally, while it is not mandatory under this license, it is considered good form to offer a free copy of any hardcopy and CD-ROM expression of an Open Publication-licensed work to its author(s).

VI. LICENSE OPTIONS

The author(s) and/or publisher of an Open Publication-licensed document may elect certain options by appending language to the reference to or copy of the license. These options are considered part of the license instance and must be included with the license (or its incorporation by reference) in derived works.

A. To prohibit distribution of substantively modified versions without the explicit permission of the author(s). "Substantive modification" is defined as a change to the semantic content of the document, and excludes mere changes in format or typographical corrections.

To accomplish this, add the phrase `Distribution of substantively modified versions of this document is prohibited without the explicit permission of the copyright holder.' to the license reference or copy.

B. To prohibit any publication of this work or derivative works in whole or in part in standard (paper) book form for commercial purposes is prohibited unless prior permission is obtained from the copyright holder.

To accomplish this, add the phrase 'Distribution of the work or derivative of the work in any standard (paper) book form is prohibited unless prior permission is obtained from the copyright holder.' to the license reference or copy.

OPEN PUBLICATION POLICY APPENDIX

(This is not considered part of the license.)

Open Publication works are available in source format via the Open Publication home page at *http://works.opencontent.org/*.

Open Publication authors who want to include their own license on Open Publication works may do so, as long as their terms are not more restrictive than the Open Publication license.

If you have questions about the Open Publication License, please contact David Wiley at dw@opencontent.org, and/or the Open Publication Authors' List at opal@opencontent.org, via email.

To **subscribe** to the Open Publication Authors' List: Send E-mail to opal-request@opencontent.org with the word "subscribe" in the body.

To **post** to the Open Publication Authors' List: Send E-mail to opal@opencontent.org or simply reply to a previous post.

To **unsubscribe** from the Open Publication Authors' List: Send E-mail to opal-request@opencontent.org with the word "unsubscribe" in the body.

Index

Packages

Programs

Libraries

Scripts

Others