Linux From Scratch

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Linux From Scratch

Version 4.0−RC1

Gerard Beekmans

Copyright © 1999−2002 by Gerard Beekmans

This book describes the process of creating a Linux system from scratch, using nothing but the sources of the required software.

Copyright (c) 1999−2002, Gerard Beekmans

All rights reserved.

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

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

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

Dedication

This book is dedicated to my loving and supportive wife *Beverly Beekmans*.

Table of Contents

Preface

Foreword

Having used a number of different Linux distributions, I was never fully satisfied with any of them. I didn't like the arrangement of the bootscripts. I didn't like the way certain programs were configured by default. Much more of that sort of thing bothered me. Finally I realized that if I wanted full satisfaction from my Linux system I would have to build my own system from scratch, using only the source code. I resolved not to use pre−compiled packages of any kind, nor CD−ROM or bootdisk that would install some basic utilities. I would use my current Linux system to develop my own.

This wild idea seemed very difficult at the time and often seemed an impossible task. After sorting out all kinds of problems, such as dependencies and compile−time errors, a custom−built Linux system was created that was fully operational. I called this system a Linux From Scratch system, or LFS for short.

I hope you will have a great time working on your own LFS!

−− Gerard Beekmans gerard@linuxfromscratch.org

Who would want to read this book

There are many reasons why somebody would want to read this book. The principle reason being to install an LFS system. 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.

One important reason for LFS' existence is to help people learn how a Linux system works from the inside out. Building an LFS system helps demonstrate what makes Linux tick, and how things work together and depend on each other. And perhaps most importantly, how to customize it to your own tastes and needs.

A key benefit of LFS is that you have more control of your system without relying on someone else's Linux implementation. With LFS, you are in the driver's seat and dictate every aspect of your system, such as the directory layout and boot script setup. You also dictate where, why and how programs are installed.

Another benefit of LFS is the ability to create a very compact Linux system. When installing a regular distribution, you end up with several programs which you are likely to never use. They're just sitting there wasting (precious) disk space. It isn't difficult to build an LFS system less than 100 MB. Does that still sound like a lot? A few of us have been working on creating a very small embedded LFS system. We successfully built a system that was just enough to run the Apache web server with approximately 8MB of disk space used. Further stripping could bring that down to 5 MB or less. Try that with a regular distribution.

We could compare distributed Linux to a hamburger you buy at a fast−food restaurant − you have no idea what you are eating. LFS, on the other hand, doesn't give you a hamburger, but the recipe to make a hamburger. This allows you to review it, to omit unwanted ingredients, and to add your own ingredients which enhance the flavor of your burger. When you are satisfied with the recipe, you go on to preparing it. You make it just the way you like it: broil it, bake it, deep−fry it, barbeque it, or eat it tar−tar (raw).

Another analogy that we can use is that of comparing LFS with a finished house. LFS will give you the skeletal plan of a house, but it's up to you to build it. You have the freedom to adjust your plans as you go.

Another advantage of a custom built Linux system is security. By compiling the entire system from source code, you are empowered to audit everything and apply all the security patches you feel are needed. You don't have to wait for somebody else to compile binary packages that fix a security hole. Unless you examine the patch and build it yourself you have no guarantee that the new package was built correctly and actually fixes the problem (adequately). You never truly know whether a security hole is fixed or not unless you do it yourself.

Who would not want to read this book

If you do not wish to build your own Linux system from scratch, then you probably don't want to read this book. Our goal is to build a complete and useable foundation system. If you only want to know what happens while your computer boots, then we recommend the "From Power Up To Bash Prompt" HOWTO. The HOWTO builds a bare system which is similar to that of this book, but it focuses strictly on creating a system capable of booting to a BASH prompt.

While you decide which to read, consider your objective. If you wish to build a Linux system while learning a bit along the way, then this book is probably your best choice. If your objective is strictly educational and you do not have any plans for your finished system, then the "From Power Up To Bash Prompt" HOWTO is probably a better choice.

The "From Power Up To Bash Prompt" HOWTO is located at http://www.netspace.net.au/~gok/power2bash/.

Organization

Much of the appendices are integrated into Part II (which enlarges the book somewhat). We believe this makes for easier reading. This way, you don't have to keep referencing an appendix while you read Part II. That's a real chore, especially if you're reading the txt version of this book. This book is divided into the following parts:

Part I – Introduction

Part I gives general information about the contents of the book (revisions, where to get it, changelog, mailing lists, and other contact information). It also contains suggested readings which discuss a few important considerations before beginning your LFS system.

Part II – Installation of the base LFS system

Part II guides you through the building and installation of an LFS system. The resulting LFS system will be the core foundation upon which the rest of your Linux system is built. Whatever your system becomes, it will be built and supported by the foundation that we build in Part II.

Part III – Appendixes

Part III contains various appendices.

I. Part I – Introduction

Table of Contents

Chapter 1. Introduction

Acknowledgments

We thank the following people and organizations for their contributions toward the Linux From Scratch project:

- Mark Stone <mstone@linux.com> for donating the linuxfromscratch.org server.
- VA Linux Systems for providing rackspace and bandwidth for the linuxfromscratch.org server.
- Fredrik Danerklint for running the se.linuxfromscratch.org mirror.
- Tim Jackson <tim@idge.net> for running the linuxfromscratch.idge.net mirror.
- Hagen Herrschaft <hrx@hrxnet.de> for running the de.linuxfromscratch.org mirrors, and for his donation of a P4−2.2GHz system to the LFS project.
- UK Mirror Service for running the linuxfromscratch.mirror.ac.uk mirror.
- Guido Passet <guido@primerelay.net> for running the www.nl.linuxfromscratch.org and ftp.snt.utwente.nl mirrors.
- Timothy Bauscher <timothy@linuxfromscratch.org> for being more than a great help in editing this book.
- Mark Hymers <markh@linuxfromscratch.org> for being more than a great help in editing this book.
- Marc Heerdink <marc_heerdink@softhome.net> for also being a great help in editing this book.
- DREAMWVR.COM for their ongoing sponsorship by donating various resources to the LFS and related sub projects.
- Jan Niemann <jan.niemann@tu.bs.de> for running the www.de.linuxfromscratch.org mirror.
- Torsten Westermann <westermann@linux−provider.net> for running the lfs.linux−provider.net mirror.
-

Ian Chilton <ian@ichilton.co.uk> for running the www.us.linuxfromscratch.org and www.linuxfromscratch.co.uk mirrors.

- Dag Stenstad <dag@stenstad.net> for providing the www.no.linuxfromscratch.org mirror, and Ian Chilton <ian@ichilton.co.uk> for running it.
- Antonin Sprinzl <Antonin.Sprinzl@tuwien.ac.at> for running the www.at.linuxfromscratch.org mirror.
- Jason Andrade <jason@dstc.edu.au> for running the www.au.linuxfromscratch.org mirror.
- Ian Cooper <ian@wpi.edu> for running the www.us2.linuxfromscratch.org mirror.
- VA Linux Systems who, on behalf of Linux.com, donated a VA Linux 420 (former StartX SP2) workstation towards this project.
- Johan Lenglet <johan@linuxfromscratch.org> for leading the French LFS translation project.
- Jesse Tie−Ten−Quee <highos@linuxfromscratch.org> for donating a Yamaha CDRW 8824E cd writer.
- O'Reilly for donating books on SQL and PHP.
- Robert Briggs for donating the linuxfromscratch.org and linuxfromscratch.com domain names.
- Frank Skettino <bkenoah@oswd.org> at OSWD for coming up with the initial design of the LFS website.
- Garrett LeSage <garrett@linux.com> for creating the LFS banner.
- Dean Benson <dean@vipersoft.co.uk> for helping out financially with setting up the LFS non−profit organization.
- Countless other people on the various LFS mailinglists who are making this book happen by giving their suggestions, testing the book and submitting bug reports.

How things are going to be done

We are going to build the LFS system by using a previously installed Linux distribution such as Debian, SuSE, Slackware, Mandrake, RedHat, etc. We will use the existing Linux system as the development platform, because we need tools like a compiler, linker, text editor, and other development tools to build our system. Ordinarily, the required tools are available by default if we selected "development" as one of our installation options when we installed a Linux distribution.

After you have downloaded the packages that make up an LFS system, we will create a new Linux native partition and filesystem. Here is where the LFS system will be compiled and installed.

The next step, Chapter 5, will discuss the installation of a number of packages that will form the basic development suite which is used to build the actual system, or needed to resolve circular dependencies. For example, you need a compiler to build a new compiler, and you need a shell in order to install a new shell. The packages in this chapter will be linked statically.

Static linking describes a method of compiling software so that it does not require the presence of libraries when building is complete. The resulting program is able to function on its own. The program is able to do so because the pieces of the program that would normally remain in the libraries are copied from the libraries and built right into the program. Ordinarily, software is built with dynamic linking. This conserves storage space and increases the efficiency of many programs. We statically link our software in Chapter 5 because we will, in theory, be moving our development system to a virtual environment where the already mentioned libraries will be absent. If the software is built dynamically, our development suite will not function. Since the libraries we are talking about are provided by our distribution Linux, the goal of Chapter 5 is to build a development environment where those libraries are not required and is therefore independent of the distribution.

In Chapter 6 we will build and install our final system. We will use the chroot program to enter a virtual environment and start a new shell whose root directory will be set to the partition where we built all the Chapter 5 software. This is very similar to rebooting and instructing the kernel to mount our LFS partition as the root partition. The reason that we don't actually reboot, but instead chroot, is that creating a bootable static system requires additional work which simply isn't necessary. As well, we can continue to use our platform system while we are building LFS. While software is being compiled and installed you can simply switch to a different VC (Virtual Console) or X desktop and continue using your computer normally.

When all the software from Chapter 6 is installed, Chapters 7, 8 and 9 will help us finalize our installation. We will set up our boot scripts in Chapter 7. In Chapter 8 we will build our final linux kernel and set up the Linux boot loader. Chapter 9 has some pointers to help you after you finish the book. Then finally, you reboot your system and boot into your new LFS system, and start to really use it.

This is the process in a nutshell. Detailed information on the steps we will take are discussed in the chapters and package descriptions as you progress through them. If something isn't completely clear now, don't worry. It should become very clear shortly.

Please read Chapter 2 carefully as it explains a few important things you should be aware of before you begin to work through Chapters 5 and later.

Conventions used in this book

To make things easy to follow, there are a number of conventions used throughout the book. Following are some examples:

```
./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/lfs/usr/info/dir'
```

This form of text (fixed width text) is showing screen output, probably as the result of commands issued, and is also used to show filenames, such as `/etc/lilo.conf`.

Emphasis

This form of text is used for several purposes in the book, mainly to emphasize important points, and to give examples of what to type.

http://www.linuxfromscratch.org/

This form of text is used for hyperlinks, both within the book and to external pages such as HowTo's, download locations, websites, etc.

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

This type of section is used mainly when creating configuration files. The first command (in bold) tells the system to create the file $LFS/etc/group from whatever is typed on the following lines until the sequence EOF is encountered. Therefore, this whole section is generally typed as seen.

Book version

This is LFS−BOOK version 4.0−RC1 dated September 28th, 2002. If this version is older than a month a newer version is probably already available for download. Check one of the mirror sites below for updated versions.

Mirror sites

Below is a list of our current HTTP and FTP mirror sites as of April 10th, 2002. This list might not be accurate anymore. The latest info can be found on our website at http://www.linuxfromscratch.org.

HTTP Mirrors

North America

- Fremont, California, USA [8 Mbit] – http://www.linuxfromscratch.org/lfs/intro.shtml
- Lufkin, Texas, USA [6 Mbit] – http://linuxfromscratch.idge.net/lfs/intro.shtml
- Columbus, Ohio, United States [1 Mbit] – http://www.us.linuxfromscratch.org/lfs/intro.shtml

Europe

- Mainz, Germany [100 Mbit] – http://lfs.linux–provider.net/lfs/intro.shtml
- Amsterdam, The Netherlands [100 Mbit] – http://www.nl.linuxfromscratch.org/lfs/intro.shtml
- Oslo, Norway [100 Mbit] – http://www.no.linuxfromscratch.org/lfs/intro.shtml
- Lancaster, United Kingdom [100 Mbit] – http://linuxfromscratch.mirror.ac.uk/lfs/intro.shtml
- Vienna Univ. of Technology, Austria [64 Mbit] – http://www.at.linuxfromscratch.org/lfs/intro.shtml
- Karlskrona, Sweden [10 Mbit] – http://www.se.linuxfromscratch.org/lfs/intro.shtml
- Freising, Germany [4 Mbit] – http://www.de.linuxfromscratch.org/lfs/intro.shtml
- Teeside, United Kingdom [256 Kbit] – http://www.linuxfromscratch.co.uk/lfs/intro.shtml

Australia

- Brisbane, Australia [155 Mbit] – http://www.au.linuxfromscratch.org/lfs/intro.shtml

FTP Mirrors

North America

- Fremont, California, USA [FTP] [8 Mbit] – ftp://ftp.linuxfromscratch.org
- Fremont, California, USA [HTTP] [8 Mbit] – http://ftp.linuxfromscratch.org
- Lufkin, Texas, USA [FTP] [6 Mbit] – ftp://linuxfromscratch.idge.net/linuxfromscratch
- Lufkin, Texas, USA [HTTP] [6 Mbit] – http://ftp.idge.net/linuxfromscratch

Europe

- Univ. of Twente, The Netherlands [HTTP] [100 Mbit] – http://ftp.snt.utwente.nl/linux/lfs/
- Univ. of Twente, The Netherlands [FTP] [100 Mbit] – ftp://ftp.snt.utwente.nl/pub/linux/lfs/
- Oslo, Norway [FTP] [100 Mbit] – ftp://ftp.no.linuxfromscratch.org/mirrors/lfs/
- Lancaster, United Kingdom [HTTP] [100 Mbit] – http://www.mirror.ac.uk/sites/ftp.linuxfromscratch.org/
- Vienna Univ. of Tech., Austria [FTP] [64 Mbit] – ftp://ftp.at.linuxfromscratch.org/pub/lfs/
- Vienna Univ. of Tech., Austria [HTTP] [64 Mbit] – http://ftp.at.linuxfromscratch.org/pub/lfs
- Karlskrona, Sweden [FTP] [10 Mbit] – ftp://ftp.se.linuxfromscratch.org/lfs
- Freising, Germany [HTTP] [4 Mbit] – http://ftp.de.linuxfromscratch.org/
- Freising, Germany [FTP] [4 Mbit] – ftp://ftp.de.linuxfromscratch.org/mirrors/ftp.linuxfromscratch.org/

Australia

- Brisbane, Australia [FTP] [155 Mbit] – ftp://ftp.planetmirror.com/pub/lfs/

Changelog

4.0−RC1 − September 28th, 2002

* Upgraded to:
    * automake−1.6.3
    * bin86−0.16.3
    * binutils−2.13
    * bison−1.35
    * diffutils−2.8.1
    * file−3.39
    * gawk−3.1.1
    * gcc−3.2
    * gettext−0.11.5
    * groff−1.18
    * gzip−1.2.4b.patch
    * lfs−bootscripts−1.10
    * linux−2.4.19
    * MAKEDEV−1.7
    *

  - man−1.5k
  - man−pages−1.52
  - modutils−2.4.19
  - ncurses−5.2−2.patch
  - perl−5.8.0
  - psmisc−21
  - texinfo−4.2
  - textutils−2.1
  - util−linux−2.11u
- Added:
  - ed−0.2.patch
  - fileutils−4.1.patch
  - gawk−3.1.1−2.patch
  - gcc−3.2.patch
  - gcc−3.2−nofixincludes.patch
  - glibc−2.2.5−2.patch
  - gzip−1.2.4b.patch
  -

- kbd−1.06−3.patch
- man−1.5k.patch
- ncurses−5.2.patch
- procps−2.0.7.patch
- sh−utils−2.0−hostname.patch
- vim−6.1.patch
- zlib−1.1.4

- Removed:
    - gzip−1.2.4a.patch
    - kbd−1.06−2.patch
    - reiserfsprogs−3.x.1b

- September 28th, 2002 [gerard]: Chapter 05 − GCC: Added the nofixincludes patch to prevent that script from running in Chapter 05. It should run in Chapter 06, so we need it to be a seperate patch.

- September 28th, 2002 [gerard]: Chapter 06 − Man: Replaced the sed contruction with a regular patch.

- September 28th, 2002 [gerard]: Chapter 06 − Bzip2: Removed the *PREFIX=/usr* from the **`make install`** command because the *PREFIX* is already set to */usr* by default.

- September 28th, 2002 [gerard]: Chapter 06 − Vim: Removed the FHS compliance notes. They are bogus because Vim doesn't use the localstatedir at all.

- September 28th, 2002 [timothy]: Applied Bill Maltby's grammatic−fixes patch. Changed "$LFS" to "LFS" when speaking of the LFS environment variable.

-

September 23rd, 2002 [timothy]: Applied Bill Maltby's grammatic–related patches.

- September 23rd, 2002 [timothy]: Added **–** before **tar** options (for clarity).

- September 22nd, 2002 [timothy]: Chapter 06: Applied Alex's grammatic–fixes patch.

- September 21st, 2002 [timothy]: Chapter 02: Applied Bill Maltby's grammatic–fixes patch.

- September 21st, 2002 [timothy]: Chapter 06 – Zlib: **mv** shared lib to /lib.

- September 20th, 2002 [timothy]: Chapter 05 – GCC: Removed the **––enable–threads=posix** flag since we don't build a C++ compiler in this chapter.

- September 18th, 2002 [timothy]: Chapter 05 – Introduction: Removed paragraph about static linking since it seems misleading and is covered in better detail in *Why Static*.

- September 18th, 2002 [timothy]: Chapter 08 – Linux: Removed **cd** command.

- September 18th, 2002 [timothy]: Chapter 06 – Ncurses: Removed old **mv /lib/*.a /usr/lib** command explanation.

- September 13th, 2002 [gerard]: Chapter 06 – Shadow: Added ––libdir=/usr/lib to the configure script options. This way a proper `libshadow.la` is generated. Also changed the **mv** command to move all the `libshadow.so*` files to the `/lib` directory. The `lib*a` files are already in the `/usr/lib` directory.

- September 13th, 2002 [gerard]: Chapter 06 – Man: Added another regex to the **sed** command that modifies the `man.conf` file. The added regex comments out the *MANPATH /usr/man* line which causes for duplicate results when using the **whatis** command

- September 13th, 2002 [gerard]: Chapter 06: Added the *Linux Threads Man Pages* installation after the Perl installation.

- September 12th, 2002 [gerard]: Chapter 06 – Create mtab link: Made the **ln** command an **ln –sf** so the existing `/etc/mtab` file, created by the **mount** command, will be removed before re–creating it as a symlink.

-

September 12th, 2002 [gerard]: Chapter 06 − Sh−utils: Added the sh−utils−hostname patch that supresses the build of the hostname program. This is done because the hostname program from the net−tools package is superior over this version.

- September 12th, 2002 [gerard]: Chapter 06 − Gawk: Updated the Gawk patch. It also changes the *DDEFPATH* directory location now.

- September 12th, 2002 [gerard]: Chapter 06 − Procps: Added patch that fixes a locale problem that makes **`top`** crash under certain locale settings.

- September 12th, 2002 [timothy]: Chapter 04 − Creating a file system: Referenced alternative filesystems in BLFS.

- September 12th, 2002 [gerard]: Removed all superfluous `/usr/lib/*.so` symbolic links from library installations.

- September 12th, 2002 [gerard]: Upgraded to lfs−bootscripts−1.10

- September 12th, 2002 [gerard]: Chapter 06 − Configure Sysvinit: Changed the sulogin line to *once* instead of having it respawn. This way it'll behave as expected (ie: a CTRL+D will continue instead of restart sulogin).

- September 12th, 2002 [gerard]: Chapter 06 − GCC: Added the *−−enable−clocale=gnu* option to ensure the proper locale mode is used by the C++ libraries.

- September 11th, 2002 [timothy]: Preface: Grammatic changes.

- September 8th, 2002 [timothy]: Chapter 06: Applied Alex's grammar changes patch.

- September 7th, 2002 [timothy]: Chapter 06 − Gzip: Added gzip−1.2.4b.patch.

- September 7th, 2002 [timothy]: Chapter 05 − Textutils: Added re_max_failures2 for old host systems.

- September 2nd, 2002 [timothy]: Chapter 06 − Bash: Removed creation of sh symlink. Creating the bash and sh symlinks: Added /bin/bash symlink, symlinked sh to bash. Gzip, Sysvinit, Util−Linux: Shortened cp command. Makedev: Removed /bin/bash creation and removal. Man: Modified sed statement to edit less call, so SGR will work.

-

September 1st, 2002 [timothy]: Chapter 06 – About debugging symbols: Removed info. about stripping /static. Man: Added sed statement to prevent groff from using SGR escape sequences.

- September 1st, 2002 [timothy]: Chapter 05 – Install all software as an unprivileged user: Added $CC='gcc –s' to omit the compilation of symbols in static packages.
- August 30th, 2002 [timothy]: Chapter 06 – Makedev: Put rm /bin/bash after device creation. Perl: Removed information about the old patch.
- August 30th, 2002 [timothy]: Chapter 05 – GCC: Re–added HAVE_GAS_HIDDEN; removed ––enable–__cxa–atexit which was incorrect and not needed in this chapter; added information about the patch.
- August 26th, 2002 [gerard]: Added a new Glibc patch and introducted a GCC patch.
- August 26th, 2002 [gerard]: Updated to automake–1.6.3, gcc–3.2, groff–1.18, makedev–1.7, perl–5.8.0, util–linux–2.11u
- August 22nd, 2002 [timothy]: Appendix: Added missing URLs to patches.
- August 18th, 2002 [timothy]: Chapter 05 & 06: Changed ln –sf to ln –s wherever possible.
- August 18th, 2002 [timothy]: Chapter 06 – Binutils: cp libiberty.h after install, as it is needed by certain software. Shadow: added command to remove the groups program installed by Shadow because Sh–utils installs a (better) groups program.
- August 18th, 2002 [timothy]: Chapter 05 – Sh–utils: Re–added sh–utils–2.0.patch.
- August 16th, 2002 [markh]: Chapter 06 – Move man–pages to just after the kernel headers installation.
- August 15th, 2002 [markh]: Chapter 06 – Move the MAKEDEV installation to before glibc and remove the temporary creation of `/dev/null` as we don't need it anymore.
- August 15th, 2002 [timothy]: Chapter 04 – Preparing a new partition: mentioned that a swap partition can be shared between the LFS and host systems, grammatic changes.
-

August 13th, 2002 [gerard]: Chapter 06: Removed the *−−with−curses* switch from the Bash installation as it's unnecessary here.

- August 9th, 2002 [timothy]: Updated to modutils−2.4.19, linux−2.4.19, gettext−0.11.5, binutils−2.13, textutils−2.1.
- August 9th, 2002 [timothy]: Chapter 06 − Vim: changed alternative editors link from hints to BLFS.
- August 8th, 2002 [gerard]: Chapter 06 − Ncurses: removed the *−−disable−termcap* configure option. Termcap is disabled by default now, so no need for this option (left over from a long time ago when it was needed).
- August 8th, 2002 [gerard]: Chapter 06 − Linux: Added the command **`cp include/asm-generic /usr/include`**. There are programs which use the files in there, as well as headers in the `asm` directory may be split up in the future, and put in the `asm−generic`.
- August 8th, 2002 [gerard]: Appendix A − Gettext: added the missing program description of msgcat.
- August 4th, 2002 [timothy]: Added zlib−1.1.4.
- August 3rd, 2002 [timothy]: Updated to man−pages−1.52, man−1.5k, gettext−0.11.4, modutils−2.4.18.
- July 29th, 2002 [timothy]: Removed Reiserfsprogs. Updated to util−linux−2.11t and file−3.39.
- July 29th, 2002 [timothy]: Chapter 04 & 05 − Creating a new partition, Introduction, Why static: grammatic changes. Diffutils, Fileutils, Grep, Texinfo: set LDFLAGS=−static before configure instead of as an argument to make. GCC: appended HAVE_GAS_HIDDEN to auto−host.h.
- July 29th, 2002 [timothy]: Chapter 06 − Glibc: added −−disable−profile flag.
- July 29th, 2002 [timothy]: Chapter 08 − Linux: added information about modules and kernel documentation.
- July 29th, 2002 [timothy]: Chapter 09 − Rebooting the system: added a command to remove the static directory.
- July 8th, 2002 [timothy]: Chapter 09 − Rebooting the system: Pointed to BLFS as the next step.
-

July 3rd, 2002 [timothy]: Chapter 06 – Sysvinit: Simplified the sed command and updated the installation description because init now prints "Sending processes" instead of "Sending all processes".

- July 2nd, 2002 [markh]: Internal change – Made all patches use a &package–patch–version; entity and removed all hardcoding of patch versions.
- June 30th, 2002 [timothy]: Updated to man–pages–1.51 and automake–1.6.2
- June 24th, 2002 [timothy]: Chapter 06 – Shadow, Util–linux, LFS–Bootscripts: Updated package contents.
- June 23rd, 2002 [timothy]: Chapter 05 & 06 – Net–tools, Perl, Texinfo, Autoconf, Automake, File, Libtool, Bin86, Vim, Linux, Bison, Less, Man–pages, Groff, Bzip2, E2fsprogs, Grep, Lilo, Modutils, Procps, Psmisc, Reiserfsprogs: Updated package contents.
- June 23rd, 2002 [timothy] Chapter 05 & 06 – M4, Bzip2, File, E2fsprogs: Added "last checked against" for uniformity. GCC: Removed i686–specific programs.
- June 16th, 2002 [timothy]: Chapter 06 – Gettext: Updated package contents.
- June 14th, 2002 [timothy]: Chapter 05 & 06 – Binutils, Bzip2, Diffutils, Grep: Updated package contents. GCC: Updated description of c++filt.
- June 13th, 2002 [timothy]: Chapter 09 – The End: Changed $LFS/etc/lfs–4.0–RC1 to $LFS/etc/lfs and put the version number inside this file.
- June 12th, 2002 [timothy]: Chapter 05 – GCC: Modified the build instructions and command explanations to only build the C compiler. The C++ compiler is not needed until after the second GCC build.
- June 12th, 2002 [timothy]: Chapter 06 – Shadow: grammatic changes.
- June 11th, 2002 [timothy]: Chapter 05 & 06 – Gawk: Created a list of package contents and descriptions. Fileutils: Removed a confusing paragraph about the fileutils patch. GCC: Updated the package contents.
- June 11th, 2002 [timothy] All software: Updated the estimated required disk space.
-

June 9th, 2002 [markh]: Chapter 06 – Creating Directories: Changed usr,usr/local to just usr/local as we use the –p option to mkdir which will create the usr directory anyways.

- June 7th, 2002 [timothy] Chapter 06 – Reiserfsprogs: added a description for unpack.
- June 7th, 2002 [timothy] Chapter 02 – How to ask for help: mentioned the FAQ.
- June 6th, 2002 [markh] – Chapter 05 – Tidy up explanations following the /static change.
- June 5th, 2002 [timothy]: Preface – Who would not want to read this book: applied a revised version of Scot's grammar patch.
- June 5th, 2002 [timothy]: Chapter 09 – Rebooting the system, Lilo, Bootscripts: named the hint authors. Chapter 06 – Vim: updated the hint URL. Chapter 05 – Gawk: to avoid confusion, mentioned that the patch will be applied in Chapter 06.
- June 3rd, 2002 [timothy] Chapter 01 – FAQ: edited to include reporting typos.
- May 31st, 2002 [gerard] Chapter 05 – Findutils: Added the CPPFLAGS...re_max_failures fix which is needed on Glibc–2.1 systems.
- May 30th, 2002 [markh]: Chapter 05 & 06 – Update to binutils–2.12.1.
- May 30th, 2002 [markh]: Chapter 05 – Bash: Removed section about "last two commands executing anyways" because we no longer have the commands referred to there.
- May 30th, 2002 [gerard]: Chapter 06 – Glibc: Replaced the various sed fixes with a regular patch.
- May 30th, 2002 [gerard]: Chapter 06 – Gawk: Replaced the sed fix with a regular patch.
- May 30th, 2002 [gerard]: Chapter 05 – Fileutils: Replaced the sed fix with a regular patch.
- May 30th, 2002 [gerard]: Chapter 06 – Ed: Replaced the sed fix with a regular patch.
- May 28th, 2002 [gerard]: Chapter 06 – Changing ownership: removed the explicit command to chown `/lost+found`. This is done by the first command now that proc isn't mounted anymore in chapter 5.
-

May 27th, 2002 [gerard]: Upgraded to ncurses−5.2−2.patch (this patch is smaller than the previously used one).

- May 26th, 2002 [gerard]: Upgraded to: automake−1.6.1, bin86−0.16.3, file−3.38, gawk−3.1.1, gcc−3.1, gettext−0.11.2, modutils−2.4.16, psmisc−21 and util−linux−2.11r. Added gcc−3.1 compile fix patches for ncurses, perl and vim.
- May 26th, 2002 [gerard]: Chapter 05+06 − Binutils: Removed the tooldir setting from chapter 05−binutils, moved its description to chapter 06−binutils.
- May 26th, 2002 [gerard]: Chapter 05 − Gawk & Findutils: simplified the installation by removing the libexecdir modifications. We can live with a `$LFS/static/libexecdir` being created. The whole `$LFS/static` directory is temporarily anyways, so we're not all that concerned with what it looks like.
- May 26th, 2002 [gerard]: Chapter 06 − Creating Directories: removed the **`cd /`** command and changed the two **`chmod`** commands to use absolute paths instead.
- May 25th, 2002 [markh]: Chapter 06 − Some minor corrections dealing with removing the $LFS variable where it isn't wanted.
- May 23rd, 2002 [gerard]: Implemented the keep_chap5_and_chap6_sep lfs−hint. Highlights of the change: added findutils and util−linux to chapter 5, installed everything from chapter 5 into `$LFS/static` and re−ordered the installation of packages in chapter 6 to prevent hard−wiring the wrong path (files from `$LFS/static`).
- May 23rd, 2002 [gerard]: Appendix A − E2fsprogs: Added some more descriptions.
- May 23rd, 2002 [gerard]: Appendix A − Bin86: Added some descriptions.
- May 23rd, 2002 [gerard]: Appendix A − Flex: Added some descriptions.
- May 23rd, 2002 [gerard]: Appendix A − Glibc: Added some more descriptions.
- May 18th, 2002 [gerard]: Appendix A − E2fsprogs: Added some descriptions.
- May 18th, 2002 [gerard]: Appendix A − Glibc: Added some more descriptions.
-

May 17th, 2002 [markh]: Changed all chown X.X's to chown X:X's which is less likely to run into problems (according to info chown).

- May 16th, 2002 [gerard]: Chapter 01 − Mirror sites: Added http interface to FTP mirror at idge.net
- May 16th, 2002 [gerard]: Appendix A − Glibc: Added some more descriptions.
- May 15th, 2002 [markh]: Chapter 05 − Bzip2. Changed the instructions to deal with hard links in older distros a'la the Chapter 05 gzip instructions.
- May 11th, 2002 [markh]: Various XML fixups; mainly altering <ulink> tags to remove erroneous
 in the HTML output.
- May 9th, 2002 [gerard]: Appendix A − Glibc: Filled in the missing descriptions.
- May 6th, 2002 [gerard]: Chapter 06 − Shadow: Fixed the symlink location of `vigr` to `/usr/sbin`
- May 2nd, 2002 [gerard]: Chapter 06 − Procps: Changed the two single quotes to two double quotes (the two single quotes can be mistaken for one double quote which will cause an error).
- May 2nd, 2002 [gerard]: Changed the **`cd dir && ln −sf`** commands to one single command (such as **`ln −sf bash $LFS/bin/sh`** Same goes for **`cd dir && mv/cp`** constructions which are now replaced with constructions like **`mv $LFS/usr/bin/{bzcat,bzip2} $LFS/bin`**
- May 2nd, 2002 [markh]: Removed the "Removing old NSS library files" section.
- May 1st, 2002 [gerard]: Removed all Glibc−2.0 workarounds − gzip patch, sh−utils patch, copying of libnss files. Also removed the **`export VAR=VALUE...unset VAR`** constructions and changed them to **`VAR=VALUE ./configure`** constructions.
- April 26th, 2002 [marcheerdink]: Chapter 06 Findutils: added libexecdir=/usr/bin to the make command to fix a wrong libexecdir path in updatedb.
- April 25th, 2002 [gerard]: Chapter 06 Glibc: added a note that if you want to manually install some locales, instead of all of them, then you first need to create the `/usr/lib/locale` directory.
- April 21st,2002 [gerard & markh]: Upgraded to MAKEDEV−1.5
-

April 12th, 2002 [markh]: Added entities/ directory to cvs and split up index.xml.

- April 10th, 2002 [marcheerdink]: Updated to the following packages: bison−1.35, diffutils−2.8.1, texinfo−4.2, util−linux−2.11q
- April 9th, 2002 [marcheerdink]: Added −−disable−perl−regexp to the grep configure flags to avoid linking against a non−existing static pcre library.
- April 8th, 2002 [gerard]: Added the http://ftp.de.linuxfromscratch.org mirror (to complement ftp://ftp.de).

3.3 − April 7th, 2002

- Updated to:
    - autoconf−2.53
    - automake−1.6
    - bin86−0.16.2
    - binutils−2.12
    - bison−1.34
    - bzip2−1.0.2
    - diffutils−2.8
    - e2fsprogs−1.27
    - gawk−3.1.0
    - gettext−0.11.1
    - grep−2.5

  - less−374
  - lfs−bootscripts−1.9
  - lilo−22.2
  - linux−2.4.18
  - man−pages−1.48
  - modutils−2.4.15
  - reiserfsprogs−3.x.1b
  - shadow−4.0.3
  - texinfo−4.1
  - util−linux−2.11o
  - vim−6.1

- April 7th, 2002 [gerard]: Added a new mirror site located in Freising, Germany
- April 5th, 2002 [gerard]: Chapter 07 − Loadkeys: Added this page explaining that you can remove the loadkeys symlink from `/etc/rc.d/rcsysinit.d` if you compiled a keymap directly into the kernel.
- April 5th, 2002 [gerard]: Chapter 06 − Configuring Keyboard: explained you can also compile the keymap directly into the kernel which has additional benefits.
- April 5th, 2002 [gerard]: Upgraded to lfs−bootscripts−1.9
- April 5th, 2002 [gerard]: Chapter 05+06 − GCC: Added commands to remove the `/usr/*-gnu` directory.
-

April 4th, 2002 [gerard]: Chapter 05 – Diffutils: Added ––disable–nls

- April 3rd, 2002 [gerard]: Appendix A – Gettext: Added the missing package descriptions.

- April 3rd, 2002 [gerard]: Chapter 05 – Mounting $LFS/proc: Added **`chown root.root $LFS/proc`**. The recursive chown operation in chapter 6 doesn't touch proc, so this'll remain owned by user *lfs*. It's not a big deal, just not a very clean thing to do.

- April 3rd, 2002 [gerard]: Chapter 06 – Groff: Added a few symlinks that are used by programs like **`xman`** and others.

- April 3rd, 2002 [gerard]: Chapter 04 – Mounting partitions: Added some notes how to deal with multiple partitions ($LFS, $LFS/usr and so on).

- April 3rd, 2002 [gerard]: Chapter 06 – E2fsprogs: Added **`install–info`** command to finish off the info page installation.

- April 3rd, 2002 [gerard]: Chapter 06 – Bzip2: Reversed the **`make`** and **`make –f Makefile–libbz2_so`**. This is needed so all object files are compiled with the PIC option (Position Independent Code).

- April 3rd, 2002 [gerard]: Chapter 05 – Linux: Shortened the installation instructions by cutting out the **`make config`** and **`make dep`** stages.

- April 1st, 2002 [gerard]: This is not a joke: Chapter 5+6 – Gawk: Added a warning to never run **`make uninstall`** on the package. It will be pretty much equivalent to **`rm –rf /usr/bin/*`** because we override the `libexec` directory definition to `/usr/bin`

- March 29th, 2002 [markh]: Chapter 05 and 06 – Updated to diffutils–2.8, modutils–2.4.15 and vim–6.1. Removed PR_PROGRAM setting for diffutils as /usr/bin/pr is now detected by the configure script. Removed sed to fix problem with shell syntax highlighting in vim as that is fixed in the new version.

- March 26th, 2002 [markh]: Chapter 02 – Asking for help: Added reference to ESR's smart–questions document.

- March 25th, 2002 [markh]: Binutils – Added libopcodes library description.

-

March 21st, 2002 [gerard]: Chapter 06 – Bzip2: Before we move /usr/bin/bzless and /usr/bin/bzmore to the /bin directory, we first remove the /bin/bzless and /bin/bzmore files. On some systems overwriting the existing files doesn't work due to hardlinks being used.

- March 21st, 2002 [gerard]: Appendix A – Sysklogd: Updated the download location to http://www.infodrom.org/projects/sysklogd/
- March 20th, 2002 [gerard]: Chapter 06 – Configure Dynamic Loader: Removed the /lib and /usr/lib directories from the ld.so.conf file. They were unnecessary.
- March 16th, 2002 [gerard]: Chapter 06+Appendix A: Removed the chroot dependencies. It's not a package so it's a bit out of place.
- March 16th, 2002 [gerard]: Chapter 05+06 – Gawk: Added commands to sed the awklib/Makefile.in file to change the *datadir* and *libexecdir* definitions
- March 15th, 2002 [gerard]: Chapter 01 – Mailing lists: Added lfs–chat description
- March 15th, 2002 [gerard]: Chapter 06–Shadow: Move libmisc.*a to /usr/lib too.
- March 14th, 2002 [gerard]: Upgraded to bison–1.34, gettext–0.11.1, grep–2.5, lfs–bootscripts–1.8, shadow–4.0.3
- March 11th, 2002 [gerard]: Upgraded to binutils–2.12
- March 11th, 2002 [gerard]: Chapter 07 – Setclock: The text here hinted towards the fact that you could skip configuring this step which isn't true unless the entire script would be removed. So the text was changed a bit to just have them create the file no matter how the hardware clock is set up.
- March 11th, 2002 [gerard]: Chapter 07 – Loadkeys: Removed the need to configure a /etc/sysconfig/keyboard file. The kbd patch makes this obsolete (loadkeys –d is used now).
- March 11th, 2002 [gerard]: Chapter 05 – Gawk: Added –Dre_max_failures=re_max_failures2 bug fix for glibc–2.1.x systems.
- March 11th, 2002 [gerard]: Chapter 06 – Bzip2: Before installing, remove /usr/bin/bz*. The bzip2 installation doesn't deal with existing files properly when making hard links, so we remove the files first.
-

March 10th, 2002 [gerard]: Chapter 06 – Configure keyboard: Added section to configure keyboard keymap file by creating the `/usr/share/kbd/keymaps/defkeymap.map.gz` symlink.

- March 9th, 2002 [gerard]: Chapter 08 – Make bootable: Added a **cp** command that finds all the kernel images from `/etc/lilo.conf` automatically and copies them to `$LFS/boot`.

- March 9th, 2002 [gerard]: Chapter 06 – Man: Moved the `man.conf` from `/usr/share/misc` to `/etc`.

- March 9th, 2002 [gerard]: Chapter 07: Added a page about the sysklogd script and explain that the default script includes the *−m 0* option to **syslogd**.

- March 8th, 2002 [gerard]: Removed the Mawk package and replaced with the Gawk package. This was done because mawk is no longer being developed, while gawk is. Mawk has some POSIX compliance bugs that are fixed in Gawk.

- March 8th, 2002 [gerard]: Updated to the following packages: autoconf−2.53, automake−1.6, bin86−0.16.2, bison−1.33, bzip2−1.0.2, e2fsprogs−1.27, gawk−3.1.0, gettext−0.11, less−374, lilo−22.2, linux−2.4.18, man−pages−1.48, modutils−2.4.14, reiserfsprogs−3.x.1b, shadow−4.0.2, texinfo−4.1, util−linux−2.11o

3.2 – March 7th, 2002

- Updated to:

  - lfs−bootscripts−1.6

- March 1st, 2002 [gerard]: Chapter 05 – Creating directories: Removed the `/usr/var` and `/usr/local/var` directories. They aren't recommended by the *FHS*.

- February 27th, 2002 [gerard]: Chapter 06 – Make: Added commands to remove the setgid kmem bit from `/usr/bin/make`. This isn't needed on Linux systems to deal with the system load and it causes some other problems too that are fixed by removing the setgid bit.

- February 26th, 2002 [gerard]: Upgraded to lfs−bootscripts−1.6

- February 17th, 2002 [gerard]: Chapter 05 – Sh−utils: Added the command again that moves $LFS/usr/bin/chroot to $LFS/usr/sbin

-

February 17th, 2002 [gerard] Updated dependencies for all packages.

- February 15th, 2002 [gerard] Chapter 01: Added a new mirror to the list located in The Netherlands (www.nl and ftp.nl).

- February 11th, 2002 [markh] Chapter 05: Sh−utils: Removed extra && from end of install instructions.

- February 10th, 2002 [gerard]: Chapter 05 − Sh−utils: Removed *su* from the *mv* command as this isn't installed in chapter 5.

3.2−RC1 − February 10th, 2002

- Updated to:
    - bison−1.31
    - file−3.37
    - glibc−2.2.5
    - kbd−1.06−2.patch
    - lfs−bootscripts−1.5
    - linux−2.4.17
    - man−pages−1.47
    - psmisc−20.2
    - sysvinit−2.84
    - util−linux−2.11n
-

February 10th, 2002 [gerard]: Chapter 6: Added a sed command to change gzexe's hardcoded /usr/bin/gzip path and change it to /bin/gzip.

- February 10th, 2002 [gerard]: Chapter 5 + 6: Moved additional programs to the ($LFS)/bin directory that are used by the bootscripts. No programs used by bootscripts (except daemons themselves) should be in the /usr directory in case /usr isn't available until far along in the boot process (when it's an NFS share for example).
- February 6th, 2002 [markh]: Appendix A − All descriptions now synced and updated.
- February 2nd, 2002 [gerard]: Chapter 6 − Changing owner: Added "cd /" so the leading slash can be removed from all the directories in the chown commands. It's more pleasant to type out this way.
- February 2nd, 2002 [gerard]: Updated to lfs−bootscripts−1.5
- February 2nd, 2002 [gerard]: Chapter 6 − Gzip: Removed the compress symlink. Gzip can uncompress .Z files but it can't compress into that format.
- February 1st, 2002 [gerard]: Updated to lfs−bootscripts−1.3
- February 1st, 2002 [gerard]: Chapter 6 − Glibc: Instead of sed'ing the `config.make` file, create the `glibc-build/configparms` file containing "cross−compiling = no".
- January 30th, 2002 [marcheerdink]: Chapters 5: Changed the commands to copy the header files to support versions of cp older than 4.1.
- January 30th, 2002 [markh]: Chapters 5+6: Added CPPFLAGS="$CPPFLAGS −D_GNU_SOURCE" to the configure command for patch. This fixes compilation on PPC and m68k platforms and doesn't hurt on x86.
- January 30th, 2002 [gerard]: Chapter 5 − Mounting proc: Rephrased the text a bit (it implied you can only mount the proc fs more than once, which isn't true anymore these days).
- January 30th, 2002 [markh]: Chapter 5: Enhanced the make mrproper explanation.
- January 30th, 2002 [marcheerdink]: Chapters 5+6: Removed the −−libexecdir flag from fileutils' configure options.
-

January 30th, 2002 [marcheerdink]: Chapters 6: Added a symlink from vipw to vigr after installing shadow.

- January 30th, 2002 [markh]: Chapters 5+6: Changed binutils and e2fsprogs installation instructions to use separate directories ala gcc and glibc.

- January 30th, 2002 [gerard]: Chapter 6 − Bootscripts: Added a chown root.root after the cp.

- January 30th, 2002 [gerard]: Appendix A − Texinfo: the info programs works on the /usr/share/info directory not /usr/doc/info.

- January 30th, 2002 [gerard]: Chapter 6 − Procps: Fixed typo the path to the app−defaults directory (it's /usr/X11R6/lib/X11/app−defaults and not usr/X11R6/lib/app−defaults).

- January 30th, 2002 [gerard]: Chapter 6 − Configure software: Simplified the commands to create the utmp, btmp, lastlog and wtmp files.

- January 30th, 2002 [gerard]: Chapter1: Moved Acknowledgements to be displayed as the first page in chapter 1.

- January 30th, 2002 [gerard]: Chapter 1: Created a separate page to list the HTTP and FTP mirrors.

- January 30th, 2002 [gerard]: Chapter 4 − Creating partition: increased the suggested partition size from 750 MB to 1 GB.

- January 29th, 2002 [gerard]: Chapter 6 − Shadow: Combined the "mv libshadow.a /usr/lib" and "mv libshadow.la /usr/lib" commands into "mv libshadow.*a /usr/lib"

- January 26th, 2002 [gerard]: Upgraded to lfs−bootscripts−1.2

- January 26th, 2002 [marcheerdink]: Chapter 6: Removed the datadir option from bisons configure flags, because recent bisons use the correct directory by default.

- January 23rd, 2002 [markh]: Chapter 6: Added the section Create /etc/mtab symlink.

- January 23rd, 2002 [gerard]: Removed the file −C command from the file installation. This package runs this command at the very end of the installation so we don't need to do this anymore.

-

January 23rd, 2002 [marcheerdink]: Chapter 4+5+6: The static environment is now built as an unprivileged user, removing the risk of overwriting files of the host distribution.

- January 22nd, 2002 [markh]: Back out linuxthreads man−page installation instructions as they don't work (they need perl which we don't have installed at that point).
- January 21st, 2002 [markh]: Updated to glibc−2.2.5. At the same time, fixed the glibc installation so that the linuxthreads man pages are installed.
- January 21st, 2002 [markh]: Updated to bison−1.31, file−3.37, kernel−2.4.17, psmisc−20.2 and sysvinit−2.84.
- January 21st, 2002 [markh]: Updated to util−linux−2.11n and removed ADD_RAW=yes as it's no longer needed.
- January 21st, 2002 [markh]: Updated to man−pages−1.47 and removed the man−pages patch.
- January 15th, 2002 [gerard]: Appendix A: Added bootscripts files (dependencies, download location, descriptions)
- January 15th, 2002 [gerard]: Chapter 6: Added bootscripts installation.
- January 15th, 2002 [gerard]: Chapter 7: Removed most of the scripts, only left the part of a few where we set up config files in /etc/sysconfig.
- January 15th, 2002 [gerard]: Chapter 6 − Configuring Sysvinit: Changed the inittab contents to match the new bootscripts.
- January 15th, 2002 [marcheerdink]: Chapter 6 − file: changed the installation instruction so the sed isn't necessary anymore.
- January 14th, 2002 [marcheerdink]: Changed the kernel header files installation in chapter 5 so it's a bit more portable.
- January 6th, 2002 [gerard]: Reformatted the dependency lists.
- January 1st, 2002 [gerard]: Happy New Year LFS!
-

January 1st, 2002 [markh]: First Changelog of New Year! Update copyright notice to cover 2002 ;−) OK − I'm sad...

- December 16th, 2001 [gerard]: Chapter 6 − Ed: Reworded why ed is optional to eliminate some confusion.

- December 16th, 2001 [gerard]: Chapter 6 − Texinfo: Reworded the TEXMF explanation to eliminate some confusion.

- December 15th, 2001 [gerard]: Chapter 4: Replaced the "One partition hint" reference with lfs_next_to_existing_systems.txt hint reference.

- December 15th, 2001 [markh]: Finish Appendix merge. All of the old appendices A, B and D are now in one (large) Appendix A.

- December 14th, 2001 [markh]: Merged appendices A and B.

- December 13th, 2001 [markh]: Appendix B: Changed dbhtml tag so that the flex page is now created as flex.html instead of flex

- December 13th, 2001 [markh]: Appendix D: Moved metalab.unc.edu and ftp.ibiblio.org references to the proper URL ibiblio.org.

- December 12th, 2001 [marcheerdink]: Chapter 6: Moved the kbd patch to the default installation instructions; upgraded to kbd−1.06−2.patch to fix installation of some programs; added the descriptions for these programs; removed the loadkeys −d warning that was a leftover from the time where loadkeys −d wasn't fixed yet.

- December 11th, 2001 [markh]: Chapter 6: Add the "why we cd $LFS before chroot" explanation.

- December 10th, 2001 [markh]: Chapter 6: Add kbd patch for loadkeys −d behaviour (patch by Matthias Benkmann; originally posted to the lfs−dev list).

- December 10th, 2001 [markh]: Chapter 6: Re−create symlinks in bash, fileutils and gcc instructions to make the Chapter 6 instructions independent of those in chapter 5.

- December 10th, 2001 [marcheerdink]: Chapter 5+6: Cleaned up the sed commands to use the backup file that was created earlier instead of writing to an intermediate 'tmp~' file.

-

December 10th, 2001 [marcheerdink]: Chapter 5+6: 'make' command for diffutils installation changed to 'make PR_PROGRAM=/usr/bin/pr.' This bug was reported by Greg Schafer.

- December 7th, 2001 [gerard]: Chapter 6: Change the configure command from *./Configure −Dprefix=/usr* to *./configure.gnu −−prefix=/usr*. This is more consistent with the installation instructions for the other packages, and the result is identical to the old way.
- December 3rd, 2001 [markh]: Chapter 2: Added the Which Platform? section.

3.1 − December 3rd, 2001

- Added:
    - reiserfsprogs−3.x.0j
- Updated to:
    - MAKEDEV−1.4
    - bash−2.05a
    - e2fsprogs−1.25
    - gettext−0.10.40
    - libtool−1.4.2
    - lilo−22.1
    - linux−2.4.16
    - man−1.5j
    - man−pages−1.43
    - modutils−2.4.12

  - sysvinit−2.83
  - util−linux−2.11m
  - vim−6.0
- November 30th, 2001 [markh]: Chapter 6: Updated to man−1.5j. Removed the sed which we had to use with the old version as the new one detects awk properly.
- November 30th, 2001 [markh]: Chapter 5: Added static library explanation originally posted on lfs−apps (when it still existed) by Plasmatic.
- November 26th, 2001 [markh]: Chapter 5+6: Updated to kernel−2.4.16 and modutils−2.4.12.
- November 26th, 2001 [markh]: Chapter 6: Added FHS compliance notes to the findutils installation.
- November 19th, 2001 [markh]: Chapter 5+6: Updated to bash−2.05a, lilo−22.1, MAKEDEV−1.4, man−pages−1.43 and util−linux−2.11m.
- November 5th, 2001 [markh]: Chapter 6: Created new lex script instead of link to flex following comment on lfs−dev. (This is similar to what we do with bison and yacc).
- October 27th, 2001 [markh]: General: Large XML Tidy−up. Shouldn't affect the book text or layout. If it does, something has gone wrong!
- October 27th, 2001 [markh]: Chapter 6: Added reiserfsprogs−3.x.0j and updated to lilo−22.0.2.
- October 24th, 2001 [markh]: General: Fixed a bundle of spelling errors which were reported.
- October 12th, 2001 [markh]: Chapter 5 − Kernel: Added explanation as to why we copy the kernel headers rather than symlink them.
- October 12th, 2001 [markh]: Appendix A − Gzip: Added uncompress to the gunzip description as it was missing.
- October 12th, 2001 [markh]: Chapter 6 − Util−linux: Removed the USRGAMES_DIR=/usr/bin entry as it's no longer needed with util−linux−2.11l.
-

October 9th, 2001 [gerard]: Chapter 6 − Kbd: Removed the −−datadir option, kbd's default is set properly already.

- October 7th, 2001 [gerard]: Chapter 6 − Shadow: Mentioned the http://hints.linuxfromscratch.org/hints/shadowpasswd_plus.txt lfs−hint
- October 7th, 2001 [gerard]: Chapter 6 − Vim: Changed the installation instructions to fix a bug in vim−6.0's `syntax/sh.vim` file, and added the CPPFLAGS variable to specify the global vimrc file as `/etc/vimrc`
- October 7th, 2001 [gerard]: Chapter 6: Updated to libtool−1.4.2, lilo−22.0, man−pages−1.40, modutils−2.4.10, sysvinit−2.83, util−linux−2.11l and vim−6.0
- October 2nd, 2001 [gerard]: Chapter 9 − The End: Added the reference to the LFS Counter at http://linuxfromscratch.org/cgi−bin/lfscounter.cgi
- September 26th, 2001 [gerard]: Chapter 1 − News server: Added reference to the news server
- September 26th, 2001 [markh]: Chapter 6 − E2fsprogs: Changed −−with−root−prefix=/ to −−with−root−prefix="" in e2fsprogs install instructions. The reason for the change is that a value of / will cause symlinks and installation paths to use things like //lib instead of just /lib. This isn't bad perse, it just doesn't look nice.
- September 26th, 2001 [markh]: Chapter 5+6: Updated to e2fsprogs−1.25, gettext−0.10.40, linux−2.4.10, modutils−2.4.9 and util−linux−2.11i.
- September 22nd, 2001 [markh]: Appendix A: Re−ordered the descriptions into alphabetical order.

3.0 − September 21st, 2001

- Updated to:
    - e2fsprogs−1.24
- September 21st, 2001 [markh]: Chapter 1+7: Changed the mailing list information to reflect the new ml structure. The Ch7 change is that the rc and rcS scripts now ask people to report problems to lfs−dev instead of lfs−discuss.
-

September 18th, 2001 [gerard]: Chapter 5+6 − GCC: Added −−enable−threads=posix to chapter 5, and changed −−enable−threads to −−enable−threads=posix in chapter 6. Although the default is posix threads when not specified, it's clearer this way what's being enabled.

- September 17th, 2001 [gerard]: Chapter 6 − Psmisc: Added notes how to deal with psmisc's pidof symlink (in case sysvinit isn't installed) and man page. Also, added −−exec−prefix=/ to psmisc's configure script in order for the programs to be installed in /bin rather than /usr/bin (bootscripts may use them, so they must be in /bin).

- September 16th, 2001 [markh]: Chapter 6 − Util−linux: Added USRGAMES_DIR=/usr/bin to the make install routine so that /usr/games isn't created for banner and it is installed in /usr/bin.

- September 14th, 2001 [markh]: Chapter 6 − E2fsprogs: Updated to version 1.24.

- September 11th, 2001 [gerard]: Chapter 6 − Man: Added missing && to 'done' and chmod the configure script to mode 755 instead of 700 (more of a default mode so people don't _have_ to be running as the owner of that file).

Mailing lists and archives

The linuxfromscratch.org server is hosting the following publicly accessible mailing lists:

- lfs−support
- lfs−dev
- lfs−announce
- lfs−security
- lfs−book
- lfs−chat
- alfs−discuss
- blfs−dev
- blfs−book
- blfs−support

lfs−support

The lfs−support mailing list provides support to users building an LFS system as far as the end of the main book. Requests for help with installing software beyond the base system should go to the blfs−support list.

lfs−dev

The lfs−dev mailing list discusses matters strictly related to the LFS−BOOK. If problems with the book come up, a bug or two need to be reported, or suggestions to improve the book should be made, this mailing list is the right one.

Requests for help should go to lfs−support or blfs−support.

lfs−announce

The lfs−announce list is a moderated list. It can be subscribed to, but you can't post any messages to this list. This list is used to announce new stable releases. The lfs−dev list will carry information about development releases as well. If a user is already on the lfs−dev list, there's little use subscribing to this list as well because everything that is posted to the lfs−announce list will be posted to the lfs−dev list as well.

lfs−security

The lfs−security mailing list discusses security−related matters. Security concerns or security problems with a package used by LFS, should be addressed on this list.

lfs−book

The lfs−book list is used by the LFS−BOOK editors to co−ordinate lfs−book's maintenance, like XML issues and the like. Actual discussion on what should be added and removed take place on lfs−dev.

lfs−chat

The lfs−chat list is a hangout place for members of the LFS Community (that includes you as well) and just chat about stuff. Doesn't even have to be computer related. Anything goes, nothing is off−topic.

alfs−discuss

The alfs−discuss list discusses the development of ALFS, which is short for Automated Linux From Scratch. The goal of this project is to develop an installation tool that installs an LFS system automatically, thus speeding up compilation by taking away the need to manually enter the commands.

blfs−dev

The blfs−dev mailing list discusses development of the BLFS−BOOK (Beyond LFS). This is the maillist to submit bug reports, and make suggestions to improve the BLFS book.

Requests for help with programs beyond the base LFS build and setup (not just those in the BLFS book) should be made in blfs−support.

blfs−book

The blfs−book list is used by the BLFS−BOOK editors to co−ordinate the maintenance of the BLFS book, such as XML source code issues and the like. Actual discussion on what should be added and removed should take place on blfs−dev.

blfs−support

The blfs−support list handles support requests for any software that is not built or installed in the LFS book. Any software beyond what is installed as part of the base LFS system can be discussed here.

Mail archives

All these lists are archived and can be viewed online at http://archive.linuxfromscratch.org/mail−archives or downloaded from http://ftp.linuxfromscratch.org/mail−archives or ftp://ftp.linuxfromscratch.org/mail−archives.

How to post to a list

You do not need to be subscribed to a mailing list in order to post to it. However, if you post to a list you're not subscribed to, make sure you mention this in your email so the list members can put you in the CC: header of an email in order for you to receive the replies.

The post address for a list is in the format of *listname@linuxfromscratch.org* where *listname* can be one of the lists in the Available lists section above. Examples of post addresses are *lfs−dev@linuxfromscratch.org, lfs−support@linuxfromscratch.org and blfs−support@linuxfromscratch.org*.

How to subscribe?

Any of the above−mentioned mailinglists can be subscribed to by sending an email to listar@linuxfromscratch.org and writing *subscribe listname* as the subject header of the message.

Multiple lists at the same time can be subscribed to by using one email. This is done by leaving the subject blank and putting all the commands in the body of the email. The email will look like:

> To: listar@linuxfromscratch.org Subject: subscribe lfs−dev subscribe blfs−support subscribe alfs−discuss

After the email is sent, the Listar program will reply with an email requesting a confirmation of the subscription request. After this confirmation email is sent back, Listar will send an email again with the message that the user has been subscribed to the list(s) along with an introduction message for that particular list.

How to unsubscribe?

To unsubscribe from a list, send an email to listar@linuxfromscratch.org and write *unsubscribe listname* as the subject header of the message.

Multiple lists can be unsubscribed at the same time using one email. This is done by leaving the subject header blank and putting all the commands in the body of the email. The email will look like:

> To: listar@linuxfromscratch.org Subject: unsubscribe lfs−dev unsubscribe blfs−support unsubscribe alfs−discuss

After the email is sent, the Listar program will reply with an email requesting a confirmation of the unsubscription request. After this confirmation email is sent back, Listar will send an email again with the message that the user has been unsubscribed from the list(s).

Other list modes

The modes that can be set by a user require sending an email to listar@linuxfromscratch.org. The modes themselves are set by writing the appropriate commands in the subject header of the message.

As the name implies, the *Set command* tells what to write to set a mode. The *Unset command* tells what to write to unset a mode.

The word "listname" in the example subject headers below should be replaced with the listname to which the mode is going to be applied. If more than one mode is to be set (to the same list or multiple lists) with one email, this can be done by leaving the subject header blank and writing all the commands in the body of the message instead.

Digests

Set command: *set listname digest* Unset command: *unset listname digest*

All lists have the digest mode available which can be set after a user has subscribed to a list. Being in digest mode will cause you to stop receiving individual messages as they are posted to the list and instead receive one email a day containing all the messages posted to the list during that day.

There is a second digest mode called digest2. When a user is set to this mode he will receive the daily digests but will also continue to receive the individual messages to the lists as they are posted. To set this mode, substitute *digest* for *digest2* in the command.

Vacation

Set command: *set listname vacation* Unset command: *unset listname vacation*

If a user is going to be away for a while or wishes to stop receiving messages from the lists but doesn't want to unsubscribe, he can change to vacation mode. This has the same effect as unsubscribing, but without having to go through the unsubscribe process and then later through the subscribe process again.

News server

All the mailing lists hosted at linuxfromscratch.org are also accessible via the NNTP server. All messages posted to a mailing list will be copied to the correspondent newsgroup, and vice versa.

The news server can be reached at *news.linuxfromscratch.org*.

FAQ

If you encounter any errors, have any questions, or if you find a typo in the book, then, please consult the FAQ (Frequently Asked Questions) page.

http://www.linuxfromscratch.org/faq/

Contact information

Please direct your emails to one of the LFS mailing lists. See Chapter 1 – Mailing lists and archives for more information on the available mailing lists.

If you need to reach Gerard Beekmans personally, send an email to gerard@linuxfromscratch.org

Chapter 2. Important information

About $LFS

Please read the following carefully: throughout this book the variable LFS will be used frequently. $LFS must at all times be replaced with the directory where the partition that contains the LFS system is mounted. How to create and where to mount the partition will be explained in full detail in Chapter 4. For example, let's assume that the LFS partition is mounted on /mnt/lfs.

When you are told to run a command like **`./configure --prefix=$LFS/static`** you actually have to execute **`./configure --prefix=/mnt/lfs/static`**.

It's important that this is done no matter where it is read; be it in commands entered in a shell, or in a file edited or created.

A possible solution is to set the environment variable LFS. This way $LFS can be entered literally instead of replacing it with /mnt/lfs. This is accomplished by running:

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

Now, if you are told to run a command such as **`./configure --prefix=$LFS/static`**, then you may type it literally. Your shell will replace $LFS with /mnt/lfs when it processes the command line (meaning when you hit enter after having typed the command).

If you plan to use $LFS, do not forget to set the LFS variable at all times. If the variable is not set and is used in a command, $LFS will be ignored and whatever is left will be executed. A command like **`echo "root:x:0:0:root:/root:/bin/bash" > $LFS/etc/passwd`** without the LFS variable set will re–create your host system's /etc/passwd file. Simply put: it will destroy your current password database file.

One way to make sure that $LFS is set at all times is adding it to the /root/.bash_profile and /root/.bashrc files so that every time you login as user root, or you **`su`** to user root, the LFS variable is set.

About SBUs

SBUs are *Static Bash Units* and they are our way of identifying how long a package takes to compile. Why don't we use normal times like anybody else?

The biggest problem is that times cannot be accurate, not even a little bit. So many people install LFS on so many different systems, the times it takes to compile something varies too much. One package may take 20 minutes on one system, but that same package may take 3 days on another (this is not an exaggeration). So instead we've come up with a *Static Bash Unit* or *SBU*.

It works like this: the very first package you compile in this book is Bash in Chapter 5 and it'll be statically linked. The time it takes to compile this package will be the basis and called the SBU. All other compile times are relative to the time it takes to install Bash. For example, GCC−3.2 takes about 9.5 SBUs and it's proven that this number is fairly consistent among a lot of different systems. So multiply 9.5 by the number of seconds it takes for Bash to install (the SBU value) and you get a close approximation of how long GCC will take on your system.

Note: We've seen that SBUs don't work well on SMP based machines. So all bets are off if you're lucky enough to have an SMP setup.

Where to store the downloaded software

Throughout this document, we will assume that all the packages that were downloaded are placed somewhere in $LFS/usr/src.

While it doesn't matter at all where you save the downloaded packages, we recommend storing it at least on the LFS partition. This just makes sense because you need to have access to those those files when you chroot to $LFS and when you boot into the LFS system, although access when booted to $LFS could be handled other ways. $LFS/usr/src is just a logical place to store source code, but by no means a requirement. You may even want to create a subdirectory under $LFS/usr/src for tarball storage. That way you can separate tarballs from temporary build directories, but again that's up to you.

The next chapter contains a list of all the packages that need to be downloaded. The LFS partition isn't created yet, so you can't store it there yet. Just save it elsewhere for now, and when the LFS partition is created, move them over.

How to install the software

Before you start using the LFS book, we should point out that all of the commands here assume that you are using the bash shell. If you aren't, the commands may work, but we can't guarantee it. If you want a simple life, use bash.

Before you can actually start doing something with a package, you need to unpack it first. Often the package files are tar'ed and gzip'ed or bzip2'ed. We're not going to write down every time how to unpack an archive. We'll explain how to do that once, in this section.

To start with, change to the $LFS/usr/src directory by running:

```
cd $LFS/usr/src
```

If a file is tar'ed and gzip'ed, it is unpacked by running either one of the following two commands, depending on the filename:

```
tar −xvzf filename.tar.gz
tar −xvzf filename.tgz
```

If a file is tar'ed and bzip2'ed, it is unpacked by running:

```
bzcat filename.tar.bz2 | tar −xv
```

Nowadays most tar programs, but not all, are patched to be able to use bzip2 files directly. They use either the −I, the −y, or the −j parameter, which work the same as the −z parameter for handling gzip files. The above construction, however, works no matter how your host system decided to patch tar.

If a file is just tar'ed, it is unpacked by running:

```
tar −xvf filename.tar
```

When an archive is unpacked, a new directory will be created under the current directory (and this book assumes that the archives are unpacked under the $LFS/usr/src directory). Please enter that new directory before continuing with the installation instructions. Again, every time this book is going to install a package, it's up to you to unpack the source archive and cd into the newly created directory.

From time to time you will be dealing with single files such as patch files. These files are generally gzip'ed or bzip2'ed. Before such files can be used they need to be uncompressed.

If a file is gzip'ed, it is unpacked by running:

```
gunzip filename.gz
```

If a file is bzip2'ed, it is unpacked by running:

```
bunzip2 filename.bz2
```

After a package has been installed, two things can be done with it: either the directory that contains the sources can be deleted, or it can be kept. We highly recommend deleting it. If you don't do this and try to re−use the same source later on in the book (for example re−using the source trees from Chapter 5 in Chapter 6), it may not work as you expect it to. Source trees from Chapter 5 will have your host distribution's settings, which don't always apply to the LFS system after you enter the chroot environment. Even running something like *make clean* doesn't always guarantee a clean source tree.

So, save yourself a lot of hassle and just remove the source directory immediately after you have installed it, but keep the downloaded tarball available for when you need it again.

There is one exception; the kernel source tree. Keep it around as you will need it later in this book when building a kernel. Nothing before then will use the kernel tree, so the source tree won't be in your way. If, however, you are short of disk space, you can remove the kernel tree and re−untar it later when required.

Which Platform?

LFS intends to be, as far as possible, platform independent. Having said that, the main LFS development work occurs on the x86 platform. We attempt to include information where possible on differences for other platforms such as PPC. If you come across a problem compiling which is not related to the x86 platform, still feel free to ask for help on the mailing lists. Even better, if you come up with a solution to a particular problem related to one of the other platforms, please let us know at the lfs−dev mailing list. We will then (subject to confirming it works) include that in the book.

How to ask for help

If you encounter a problem while using this book, and your problem is not listed in the FAQ, you will find that most of the people on Internet Relay Chat (IRC) and on the mailing lists are willing to help you. An overview of the LFS mailing lists can be found in Chapter 1 − Mailing lists and archives. To assist us in diagnosing and solving your problem, include as much relevant information as possible in your request for help.

Things to mention

Apart from a brief explanation of the problem you're having, the essential things to include in your request are:

- the version of the book you are using (being 4.0−RC1),
- the package or section giving you problems,
- the exact error message or symptom you are receiving,
- whether you have deviated from the book at all.

(Note that saying that you've deviated from the book doesn't mean that we won't help you. After all, LFS is about choice. It'll just help us to see other possible causes of your problem.)

Configure problems

When something goes wrong during the stage where the configure script is run, look at the last lines of the `config.log`. This file may contain errors encountered during configure which weren't printed to the screen. Include those relevant lines if you decide to ask for help.

Compile problems

To help us find the cause of the problem, both screen output and the contents of various files are useful. The screen output from both the ./configure script and the make run can be useful. Don't blindly include the whole thing but on the other hand, don't include too little. As an example, here is some screen output from make:

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

```
make[2]: *** [make] Error 1
make[2]: Leaving directory `/lfs/tmp/make-3.79.1'
make[1]: *** [all-recursive] Error 1
make[1]: Leaving directory `/lfs/tmp/make-3.79.1'
make: *** [all-recursive-am] Error 2
```

In this case, many people just include the bottom section where it says

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

and onwards. This isn't enough for us to diagnose the problem because it only tells us that *something* went wrong, not *what* went wrong. The whole section, as in the example above, is what should be included to be helpful, because it includes the command that was executed and the command's error message(s).

An excellent article on asking for help on the Internet in general has been written by Eric S. Raymond. It is available online at http://www.tuxedo.org/~esr/faqs/smart–questions.html. Read and follow the hints in that document and you are much more likely to get a response to start with and also to get the help you actually need.

II. Part II – Installing the LFS system

Table of Contents

Chapter 3. Packages that need to be downloaded

Introduction

Below is a list of packages you need to download 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 upon them.

All the URL's below refer to the main LFS server. There are several FTP mirrors available from which you can download the files as well. The addresses of these can be found in Chapter 1 − Mirror sites.

The LFS FTP archive only contains the versions of the packages that are used in this book. You can check the official download sites provided in Appendix A to determine whether or not a newer package is available. If you do download a newer package, we would appreciate hearing whether you were able to install the package without any problems using this book's instructions.

Packages that need to be downloaded

Browse FTP: ftp://ftp.linuxfromscratch.org/ Browse HTTP: http://ftp.linuxfromscratch.org/
You can either download one tarball that contains all the packages
used to compile an LFS system: All LFS Packages − 105,560 KB:
ftp://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/lfs−packages−4.0−rc1.tar
http://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/lfs−packages−4.0−rc1.tar
Or download the following packages individually: Autoconf (2.53) − 739 KB:
ftp://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/autoconf−2.53.tar.bz2
http://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/autoconf−2.53.tar.bz2 Automake (1.6.3) − 465 KB:
ftp://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/automake−1.6.3.tar.bz2
http://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/automake−1.6.3.tar.bz2 Bash (2.05a) − 1,400 KB:
ftp://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/bash−2.05a.tar.bz2
http://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/bash−2.05a.tar.bz2 Bin86 (0.16.3) − 113 KB:
ftp://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/bin86−0.16.3.tar.bz2
http://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/bin86−0.16.3.tar.bz2 Binutils (2.13) − 9,651 KB:
ftp://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/binutils−2.13.tar.bz2
http://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/binutils−2.13.tar.bz2 Bison (1.35) − 613 KB:
ftp://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/bison−1.35.tar.bz2
http://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/bison−1.35.tar.bz2 Bzip2 (1.0.2) − 610 KB:
ftp://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/bzip2−1.0.2.tar.bz2
http://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/bzip2−1.0.2.tar.bz2 Diffutils (2.8.1) − 642 KB:
ftp://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/diffutils−2.8.1.tar.bz2
http://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/diffutils−2.8.1.tar.bz2 E2fsprogs (1.27) − 1,176 KB:
ftp://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/e2fsprogs−1.27.tar.bz2
http://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/e2fsprogs−1.27.tar.bz2 Ed (0.2) − 158 KB:
ftp://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/ed−0.2.tar.bz2
http://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/ed−0.2.tar.bz2 Ed Patch (0.2) − 1 KB:
ftp://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/ed−0.2.patch.bz2
http://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/ed−0.2.patch.bz2 File (3.39) − 151 KB:
ftp://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/file−3.39.tar.bz2
http://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/file−3.39.tar.bz2 Fileutils (4.1) − 1,217 KB:
ftp://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/fileutils−4.1.tar.bz2
http://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/fileutils−4.1.tar.bz2 Fileutils Patch (4.1) − 1 KB:
ftp://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/fileutils−4.1.patch.bz2
http://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/fileutils−4.1.patch.bz2 Findutils (4.1) − 226 KB:
ftp://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/findutils−4.1.tar.bz2
http://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/findutils−4.1.tar.bz2 Findutils Patch (4.1) − 1 KB:
ftp://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/findutils−4.1.patch.bz2
http://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/findutils−4.1.patch.bz2 Flex (2.5.4a) − 278 KB:
ftp://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/flex−2.5.4a.tar.bz2
http://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/flex−2.5.4a.tar.bz2 Gawk (3.1.1) − 1,420 KB:
ftp://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/gawk−3.1.1.tar.bz2
http://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/gawk−3.1.1.tar.bz2 Gawk Patch (3.1.1−2) − 1 KB:
ftp://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/gawk−3.1.1−2.patch.bz2
http://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/gawk−3.1.1−2.patch.bz2 GCC (3.2) − 20,043 KB:
ftp://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/gcc−3.2.tar.bz2
http://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/gcc−3.2.tar.bz2 GCC Patch (3.2) − 4 KB:
ftp://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/gcc−3.2.patch.bz2
http://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/gcc−3.2.patch.bz2 GCC nofixincludes Patch:

ftp://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/gcc−3.2−nofixincludes.patch.bz2
http://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/gcc−3.2−nofixincludes.patch.bz2
Gettext (0.11.5) − 2,489 KB: ftp://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/gettext−0.11.5.tar.bz2
http://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/gettext−0.11.5.tar.bz2 Glibc (2.2.5) − 12,114 KB:
ftp://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/glibc−2.2.5.tar.bz2
http://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/glibc−2.2.5.tar.bz2 Glibc Patch (2.2.5−2) − 8 KB:
ftp://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/glibc−2.2.5−2.patch.bz2
http://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/glibc−2.2.5−2.patch.bz2
Glibc−linuxthreads (2.2.5) − 164 KB:
ftp://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/glibc−linuxthreads−2.2.5.tar.bz2
http://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/glibc−linuxthreads−2.2.5.tar.bz2
Grep (2.5) − 545 KB: ftp://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/grep−2.5.tar.bz2
http://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/grep−2.5.tar.bz2 Groff (1.18) − 1,739 KB:
ftp://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/groff−1.18.tar.bz2
http://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/groff−1.18.tar.bz2 Gzip (1.2.4a) − 179 KB:
ftp://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/gzip−1.2.4a.tar.bz2
http://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/gzip−1.2.4a.tar.bz2 Gzip Patch (1.2.4b) − 1 KB:
ftp://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/gzip−1.2.4b.patch.bz2
http://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/gzip−1.2.4b.patch.bz2 Kbd (1.06) − 559 KB:
ftp://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/kbd−1.06.tar.bz2
http://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/kbd−1.06.tar.bz2 Kbd Patch (1.06−3) − 3 KB:
ftp://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/kbd−1.06−3.patch.bz2
http://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/kbd−1.06−3.patch.bz2 Less (374) − 189 KB:
ftp://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/less−374.tar.bz2
http://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/less−374.tar.bz2 LFS−Bootscripts (1.10) − 27 KB:
ftp://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/lfs−bootscripts−1.10.tar.bz2
http://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/lfs−bootscripts−1.10.tar.bz2
Libtool (1.4.2) − 653 KB: ftp://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/libtool−1.4.2.tar.bz2
http://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/libtool−1.4.2.tar.bz2 Lilo (22.2) − 292 KB:
ftp://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/lilo−22.2.tar.bz2
http://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/lilo−22.2.tar.bz2 Linux (2.4.19) − 25,432 KB:
ftp://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/linux−2.4.19.tar.bz2
http://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/linux−2.4.19.tar.bz2 M4 (1.4) − 249 KB:
ftp://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/m4−1.4.tar.bz2
http://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/m4−1.4.tar.bz2 Make (3.79.1) − 794 KB:
ftp://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/make−3.79.1.tar.bz2
http://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/make−3.79.1.tar.bz2 MAKEDEV (1.7) − 8 KB:
ftp://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/MAKEDEV−1.7.bz2
http://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/MAKEDEV−1.7.bz2 Man (1.5k) − 168 KB:
ftp://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/man−1.5k.tar.bz2
http://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/man−1.5k.tar.bz2 Man Patch (1.5k) − 1 KB
ftp://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/kbd−1.06−3.patch.bz2
http://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/kbd−1.06−3.patch.bz2 Man−pages (1.52) − 569 KB:
ftp://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/man−pages−1.52.tar.bz2
http://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/man−pages−1.52.tar.bz2
Modutils (2.4.19) − 213 KB: ftp://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/modutils−2.4.19.tar.bz2
http://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/modutils−2.4.19.tar.bz2 Ncurses (5.2) − 1,308 KB:
ftp://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/ncurses−5.2.tar.bz2
http://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/ncurses−5.2.tar.bz2 Ncurses Patch (5.2) − 1 KB:
ftp://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/ncurses−5.2−2.patch.bz2
http://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/ncurses−5.2−2.patch.bz2

Netkit−base (0.17) − 49 KB: ftp://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/netkit−base−0.17.tar.bz2 http://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/netkit−base−0.17.tar.bz2 Net−tools (1.60) − 194 KB: ftp://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/net−tools−1.60.tar.bz2 http://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/net−tools−1.60.tar.bz2 Patch (2.5.4) − 149 KB: ftp://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/patch−2.5.4.tar.bz2 http://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/patch−2.5.4.tar.bz2 Perl (5.8.0) − 8,416 KB: ftp://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/perl−5.8.0.tar.bz2 http://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/perl−5.8.0.tar.bz2 Procinfo (18) − 22 KB: ftp://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/procinfo−18.tar.bz2 http://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/procinfo−18.tar.bz2 Procps (2.0.7) − 153 KB: ftp://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/procps−2.0.7.tar.bz2 http://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/procps−2.0.7.tar.bz2 Procps Patch (2.0.7) − 1 KB: ftp://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/procps−2.0.7.patch.bz2 http://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/procps−2.0.7.patch.bz2 Psmisc (21) − 172 KB: ftp://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/psmisc−21.tar.bz2 http://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/psmisc−21.tar.bz2 Sed (3.02) − 221 KB: ftp://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/sed−3.02.tar.bz2 http://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/sed−3.02.tar.bz2 Shadow (4.0.3) − 760 KB: ftp://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/shadow−4.0.3.tar.bz2 http://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/shadow−4.0.3.tar.bz2 Sh−utils (2.0) − 824 KB: ftp://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/sh−utils−2.0.tar.bz2 http://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/sh−utils−2.0.tar.bz2 Sh−utils Hostname Patch (2.0−hostname) − 1 KB: ftp://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/sh−utils−2.0−hostname.patch.bz2 http://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/sh−utils−2.0−hostname.patch.bz2 Sh−utils Patch (2.0) − 1 KB: ftp://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/sh−utils−2.0.patch.bz2 http://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/sh−utils−2.0.patch.bz2 Sysklogd (1.4.1) − 67 KB: ftp://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/sysklogd−1.4.1.tar.bz2 http://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/sysklogd−1.4.1.tar.bz2 Sysvinit (2.84) − 76 KB: ftp://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/sysvinit−2.84.tar.bz2 http://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/sysvinit−2.84.tar.bz2 Tar (1.13) − 730 KB: ftp://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/tar−1.13.tar.bz2 http://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/tar−1.13.tar.bz2 Tar Patch (1.13) − 1 KB: ftp://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/tar−1.13.patch.bz2 http://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/tar−1.13.patch.bz2 Texinfo (4.2) − 1,175 KB: ftp://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/texinfo−4.2.tar.bz2 http://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/texinfo−4.2.tar.bz2 Textutils (2.1) − 1,847 KB: ftp://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/textutils−2.1.tar.bz2 http://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/textutils−2.1.tar.bz2 Util−linux (2.11u) − 1,073 KB: ftp://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/util−linux−2.11u.tar.bz2 http://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/util−linux−2.11u.tar.bz2 Vim (6.1) − 2,823 KB: ftp://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/vim−6.1.tar.bz2 http://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/vim−6.1.tar.bz2 Vim Patch (6.1) − 1 KB: ftp://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/vim−6.1.patch.bz2 http://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/vim−6.1.patch.bz2 Zlib (1.1.4) − 144 KB: ftp://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/zlib−1.1.4.tar.bz2 http://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/zlib−1.1.4.tar.bz2
Total size of all packages: 105,560 KB (103.88 MB)

Chapter 4. Preparing a new partition

Introduction

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

Creating a new partition

It is possible to build LFS on only one partition – the partition in which your original distribution is installed. While this is not recommended for your first LFS installation, it may be useful if you are short on disk space. If you feel brave, take a look at the *Install LFS next to existing systems on the same partition* hint at http://hints.linuxfromscratch.org/hints/lfs_next_to_existing_systems.txt.

Before we can build our new Linux system, we need an empty Linux partition where we can build it. We recommend a partition size of at least 1 GB. This provides enough space to store the tarballs and compile all of the packages. You will probably need more space if you intend to install additional software and use the LFS system as your primary Linux system. If a Linux native partition is already available, this subsection can be skipped.

Since your system memory can only hold a limited amount of data at one time, we recommend that disk space be set aside for swap files. A swap file is a place where items in memory may be stored until they are called for. This disk space may be shared between your host system and your LFS system. If you already have a swap partition, then you probably don't need to create another one. Otherwise, you should create a swap partition via an fdisk program. Regardless, you need to remember the designation of the swap partition (such as hda2) as it will be needed when we create the `/etc/fstab` file.

The cfdisk program (or another fdisk–like program) should be started with the appropriate hard disk as the argument (like /dev/hda if a new partition is to be created on the primary master IDE disk). Using this program, create a Linux native partition. Please refer to the documentation of your fdisk program (the man pages are often a good place to start) for information about creating Linux native partitions and writing partition tables.

The designation of your new partition should be remembered. It might be something similar to hda11. This newly created partition will be referred to as the LFS partition in this book.

Creating a file system on the new partition

Once the partition is created, we have to create a new file system on that partition. The standard file system used these days is the ext2 file system, but the so–called journaling file systems are becoming increasingly popular too. We'll assume that you wish to create an ext2 file system. However, build instructions for other file systems may be found at http://beyond.linuxfromscratch.org/view/cvs/postlfs/filesystems.html.

To create an ext2 file system, use the mke2fs command. The LFS partition is used as the only option to the command and the file system is created.

```
mke2fs /dev/xxx
```

Replace "xxx" by the partition's designation (like hda11).

Mounting the new partition

Now that we have created a file system, it is ready for use. All we have to do to be able to access the partition (as in reading data from and writing data to) is mount it. If it is mounted under /mnt/lfs, this partition can be accessed by cd'ing to the /mnt/lfs directory. This book will assume that the partition was mounted under /mnt/lfs. It doesn't matter which directory is chosen, just make sure you remember what you chose.

Create the /mnt/lfs directory by running:

```
mkdir -p /mnt/lfs
```

Now mount the LFS partition by running:

```
mount /dev/xxx /mnt/lfs
```

Replace "xxx" by the partition's designation (like hda11).

This directory (/mnt/lfs) is the LFS variable you have read about back in Chapter 2. If you were planning to make use of the LFS environment variable, **`export LFS=/mnt/lfs`** has to be executed now.

If you decided to create multiple partitions for LFS (say $LFS and $LFS/usr), mount them like this:

```
mkdir -p /mnt/lfs &&
mount /dev/xxx /mnt/lfs &&
mkdir /mnt/lfs/usr &&
mount /dev/yyy /mnt/lfs/usr
```

Of course, replace /dev/xxx and /dev/yyy with the appropriate partition designations.

Chapter 5. Preparing the LFS system

Introduction

In this chapter we will compile and install a minimal Linux system. This system will contain just enough tools to be able to start constructing the final LFS system in the next chapter.

The files compiled in this chapter will be installed under the `$LFS/static` directory, to keep them separate from the files installed in the next chapter. Since the packages compiled here are merely temporary, we don't want them to pollute the soon−to−be LFS system.

The key to learning what makes a Linux system work is to know exactly what each package is used for, and why the user or the system needs it. For this purpose a short description of the content of each package is given right after the installation instructions.

Many of our packages must be patched before they can be compiled. We only apply patches when and where they are needed. So, don't fret if it seems like instructions for a patch are missing.

During the installation of several packages you will probably see all kinds of compiler warnings scroll by on your screen. These are normal and can be safely ignored. They are just what they say they are: warnings −− mostly about improper, but not illegal, use of the C or C++ syntax. It's just that C standards have changed rather often and some packages still use the older standard, which is not really a problem.

Before you start, make sure the LFS environment variable is set up properly if you decided to make use of it. Run the following:

```
echo $LFS
```

Check to make sure the output contains the correct directory to the LFS partition's mount point (/mnt/lfs for example).

Why do we use static linking?

(Thanks to Plasmatic for posting the text on which this is mainly based to one of the LFS mailing lists.)

When making (compiling) a program, rather than having to rewrite all the functions for dealing with the kernel, hardware, files, etc. every time you write a new program, all these basic functions are instead kept in libraries. glibc, which you install later, is one of these major libraries, which contains code for all the basic functions programs use, like opening files, printing information on the screen, and getting feedback from the user. When the program is compiled, these libraries of code are linked together with the new program, so that it can use any of the functions that the library has.

However, these libraries can be very large (for example, libc.a can often be around 2.5 MB), so you may not want a separate copy of each library attached to the program. Just imagine if you had a simple command like ls with an extra 2.5 MB attached to it! Instead of making the library an actual part of the program, or statically linked, the library is stored as a separate file, which is loaded only when the program needs it. This is what we call dynamically linked, as the library is loaded and unloaded dynamically, as the program needs it.

So now we have a 1 KB file and a 2.5 MB file, but we still haven't saved any space (except maybe RAM until the library is needed). The *real* advantage of dynamically linked libraries is that we only need one copy of the library. If `ls` and `rm` both use the same library, then we don't need two copies of the library, as they can both get the code from the same file. Even when in memory, the two programs share the same code, rather than loading duplicates into memory. So not only are we saving hard disk space, but also precious RAM.

If dynamic linking saves so much room, then why are we making everything statically linked? Well, that's because when you chroot into your brand new (but very incomplete) LFS environment, these dynamic libraries won't be available because they are somewhere else in your old directory tree (`/usr/lib` for example) which won't be accessible from within your LFS root (`$LFS`).

So in order for your new programs to run inside the chroot environment you need to make sure that the libraries are statically linked when you build them, hence the **`--enable-static-link`**, **`--disable-shared`**, and **`-static`** flags used through Chapter 5. Once in Chapter 6, the first thing we do is build the main set of system libraries, glibc. Once this is made we start rebuilding all the programs we just did in Chapter 5, but this time dynamically linked, so that we can take advantage of the space saving opportunities.

And there you have it, that's why you need to use those weird **`-static`** flags. If you try building everything without them, you'll see very quickly what happens when you chroot into your newly crippled LFS system.

If you want to know more about Dynamically Linked Libraries, consult a book or website on programming, especially a Linux–related site.

Creating the $LFS/static directory

As explained in this chapter's introduction, everything we install from this chapter will be installed under the `$LFS/static` directory. This way it won't pollute the LFS partition with a bunch of temporary files. All we need to do is create this directory so we can start installing. Simply run this command to create the directory:

```
mkdir $LFS/static
```

You may want to move the packages you downloaded in Chapter 3 to this `$LFS/static` directory, perhaps create a subdirectory `$LFS/static/src` to keep them in.

Install all software as an unprivileged user

When you are logged in as root during Chapter 5, it is possible that some files of your host system will be overwritten by the ones you'll build in Chapter 5. There can be all kinds of reasons for this to happen, for example because the $LFS environment variable is not set. Overwriting some files from your host system will most likely cause all kinds of problems, so it's a good idea to be logged in as an unprivileged user during Chapter 5. To make sure the environment is as clean as possible, we'll create a new user "lfs" that can be used while building the static installation. Issuing the following commands as root will create a new user "lfs":

```
useradd -s /bin/bash -m lfs &&
passwd lfs
```

Now we need to give proper permissions to the `$LFS/static` directory so user "lfs" can write to it:

```
chown -R lfs $LFS/static
```

Now you can login as user "lfs". You can do this two ways: either the normal way through the console or the display manager, or with **`su - lfs`**. When you're working as user "lfs", type the following commands to set up a good environment to work in:

```
cat > ~/.bash_profile << "EOF"
umask 022

LFS=/mnt/lfs
LC_ALL=POSIX
CC='gcc -s'
export LFS LC_ALL CC
EOF
source ~/.bash_profile
```

This profile makes sure the umask is set to 022 so newly created files and directories will have the correct permissions. It is advisable to keep this setting throughout your LFS installation. Also, the $LFS, $LC_ALL, and $CC environment variables are set. $LFS has been explained in previous chapters already. $LC_ALL is a variable that is used for internationalization.

When your host distribution uses a glibc version older than 2.2.4, having $LC_ALL set to something other than "C" or "POSIX" while working through Chapter 5 may cause trouble when you exit the chroot environment of Chapter 6 and try to return to it. By setting this to "POSIX" ("C" is an alias for "POSIX") we ensure that everything will work as expected in the chroot environment.

$CC is a variable we set in order to prevent debugging symbols from being compiled into our static packages. By omitting these symbols during the linking stage of compilation, we save hard drive space and decrease our build time.

Installing Bash−2.05a

```
Estimated build time:           1 SBU
Estimated required disk space:  24 MB
```

Installation of Bash

Before you attempt to install Bash, you have to check to make sure your distribution has the `/usr/lib/libcurses.a` and `/usr/lib/libncurses.a` files. If your host distribution is an LFS system, all files will be present if you followed the instructions of the book version you read exactly.

If both of the files are missing, you have to install the Ncurses development package. This package is often called something like *ncurses−dev*. If this package is already installed, or you just installed it, check for the two files again. Often the `libcurses.a` file is (still) missing. If so, then create `libcurses.a` as a symlink by running the following commands as user root:

```
ln -s libncurses.a /usr/lib/libcurses.a
```

Now we can continue. Install Bash by running the following commands:

```
./configure --enable-static-link \
    --prefix=$LFS/static --with-curses &&
make &&
make install
```

If the **`make install`** phase ends with something along the lines of this:

```
install-info: unknown option `--dir-file=/mnt/lfs/usr/info/dir'
usage: install-info [--version] [--help] [--debug] [--maxwidth=nnn]
        [--section regexp title] [--infodir=xxx] [--align=nnn]
        [--calign=nnn] [--quiet] [--menuentry=xxx]
        [--info-dir=xxx]
        [--keep-old] [--description=xxx] [--test]
        [--remove] [--] filename
make[1]: *** [install] Error 1
make[1]: Leaving directory `/mnt/lfs/usr/src/bash-2.05a/doc'
make: [install] Error 2 (ignored)
```

then that means that you are probably using Debian, and that you have an old version of the texinfo package. This error is not severe by any means: the info pages will be installed when we recompile bash dynamically in Chapter 6, so you can ignore it.

Command explanations

`--enable-static-link:` This configure option causes bash to be linked statically.

`--prefix=$LFS/static:` This configure option installs all of Bash's files under the $LFS/static directory, which becomes the /static directory when chroot'ed or reboot'ed into LFS.

`--with-curses:` This causes bash to be linked against the curses library instead of the default termcap library which is becoming obsolete.

It is not strictly necessary for the static bash to be linked against libncurses (it can link against a static termcap for the time being just fine because we will reinstall Bash in Chapter 6 anyways, where we will use libncurses), but it's a good test to make sure that the Ncurses package has been installed properly. If not, you will get in trouble later on in this chapter when you install the Texinfo package. That package requires ncurses, and termcap can't reliably be used there.

The **&&**'s at the end of every line cause the next command to be executed only if the previous command exits with a return value of 0 indicating success. In case all of these commands are copy & pasted on the shell, it is important to ensure that if ./configure fails, make isn't executed and, likewise, if make fails, that make install isn't executed, and so forth.

Contents of Bash

Last checked against version 2.05a.

Program Files

bash, sh (link to bash) and bashbug

Descriptions

bash

bash is the Bourne−Again SHell, which is a widely used command interpreter on Unix systems. The bash program reads from standard input, the keyboard. A user types something and the program will evaluate what he has typed and do something with it, like running a program.

bashbug

bashbug is a shell script to help the user compose and mail bug reports concerning bash in a standard format.

sh

sh is 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.

Bash Installation Dependencies

Last checked against version 2.05a.

Bash: bash, sh Binutils: ar, as, ld, ranlib, size Diffutils: cmp
Fileutils: chmod, cp, install, ln, ls, mkdir, mv, rm Gcc: cc, cc1, collect2, cpp0, gcc
Grep: egrep, grep Make: make Gawk: awk Sed: sed
Sh−utils: basename, echo, expr, hostname, sleep, uname Texinfo: install−info Textutils: cat, tr, uniq

Installing Binutils−2.13

```
Estimated build time:           2.05 SBU
Estimated required disk space:  160 MB
```

Installation of Binutils

This package is known to behave badly when you have changed its default optimization flags (including the −march and −mcpu options). Binutils is best left alone. Therefore, if you have defined any environment variables that override default optimizations, such as CFLAGS and CXXFLAGS, we recommend unsetting or modifying them when building binutils. You have been warned.

Install Binutils by running the following commands:

```
mkdir ../binutils-build &&
cd ../binutils-build &&
../binutils-2.13/configure --prefix=$LFS/static --disable-nls &&
make LDFLAGS=-all-static &&
make install
```

Command explanations

mkdir ../binutils−build: The installation instructions for Binutils recommend creating a separate build directory instead of compiling the package inside the source tree. So, we create a binutils−build directory and work from there.

−−disable−nls: This option disables internationalization (also known as i18n). We don't need this for our static programs and nls often causes problems when you're linking statically.

LDFLAGS=−all−static: Setting the variable LDFLAGS to the value −all−static causes binutils to be linked statically.

Contents of Binutils

Last checked against version 2.12.1.

Program Files

addr2line, ar, as, gasp, gprof, ld, nm, objcopy, objdump, ranlib, readelf, size, strings and strip

Descriptions

addr2line

addr2line translates program addresses into file names and line numbers. Given an address and an executable, it uses the debugging information in the executable to figure out which file name and line number are associated with a given address.

ar

The ar program creates, modifies, and extracts from archives. An archive is a single file holding a collection of other files in a structure that makes it possible to retrieve the original individual files (called members of the archive).

as

as is primarily intended to assemble the output of the GNU C compiler, gcc, for use by the linker ld.

gasp

gasp is the Assembler Macro Preprocessor.

gprof

gprof displays call graph profile data.

ld

ld combines a number of object and archive files, relocates their data and ties up symbol references. Often the last step in building a new compiled program to run is a call to ld.

nm

nm lists the symbols from object files.

objcopy

objcopy utility copies the contents of an object file to another. objcopy uses the GNU BFD Library to read and write the object files. It can write the destination object file in a format different from that of the source object file.

objdump

objdump displays information about one or more object files. The options control what particular information to display. This information is mostly useful to programmers who are working on the compilation tools, as opposed to programmers who just want their program to compile and work.

ranlib

ranlib generates an index to the contents of an archive, and stores it in the archive. The index lists each symbol defined by an archive member that is a relocatable object file.

readelf

readelf displays information about elf type binaries.

size

size lists the section sizes −−and the total size−− for each of the object files in its argument list. By default, one line of output is generated for each object file or each module in an archive.

strings

For each file given, strings prints the printable character sequences that are at least 4 characters long (or the number specified with an option to the program) and are followed by an unprintable character. By default, it only prints the strings from the initialized and loaded sections of object files. For other types of files, it prints the strings from the whole file.

strings is mainly useful for determining the contents of non−text files.

strip

strip discards all or specific symbols from object files. The list of object files may include archives. At least one object file must be given. strip modifies the files named in its argument, rather than writing modified copies under different names.

Library Files

libbfd.[a,so] and libopcodes.[a,so]

Descriptions

libbfd

libbfd is the Binary File Descriptor library.

libopcodes

libopcodes is a native library for dealing with opcodes and is used in the course of building utilities such as objdump. Opcodes are actually "readable text" versions of instructions for the processor.

Binutils Installation Dependencies

Last checked against version 2.11.2.

Autoconf: autoconf, autoheader Automake: aclocal, automake Bash: sh
Binutils: ar, as, ld, nm, ranlib, strip Diffutils: cmp
Fileutils: chmod, cp, ln, ls, mkdir, mv, rm, rmdir, touch Flex: flex
Gcc: cc, cc1, collect2, cpp0, gcc Glibc: ldconfig Grep: egrep, fgrep, grep M4: m4 Make: make
Gawk: gawk Sed: sed Sh−utils: basename, echo, expr, hostname, sleep, true, uname
Texinfo: install−info, makeinfo Textutils: cat, sort, tr, uniq

Installing Bzip2−1.0.2

```
Estimated build time:           0.07 SBU
Estimated required disk space:  6 MB
```

Installation of Bzip2

Install Bzip2 by running the following commands:

```
make CC="gcc −static" &&
make PREFIX=$LFS/static install
```

Although it's not strictly a part of a basic LFS system it's worth mentioning that a patch for Tar can be downloaded which enables the tar program to compress and uncompress using bzip2/bunzip2 easily. With a plain tar, you have to use constructions like **`bzcat file.tar.bz | tar −xv`** or **`tar −−use−compress−prog=bunzip2 −xvf file.tar.bz2`** to use bzip2 and bunzip2 with tar. This patch provides the **`−j`** option so you can unpack a bzip2'ed archive with **`tar −xvfj file.tar.bz2`**. Applying this patch will be mentioned later on when the Tar package is installed.

Command explanations

`make CC="gcc −static":` This is the method we use to tell gcc that we want bzip2 to be linked statically.

Contents of Bzip2

Last checked against version 1.0.2

Program Files

bunzip2 (link to bzip2), bzcat (link to bzip2), bzcmp, bzdiff, bzegrep, bzfgrep, bzgrep, bzip2, bzip2recover, bzless and bzmore

Descriptions

bunzip2

bunzip2 decompresses files that are compressed with bzip2.

bzcat

bzcat (or bzip2 −dc) decompresses all specified files to the standard output.

bzcmp, bzdiff

bzcmp and bzdiff are used to invoke the cmp or the diff program on bzip2 compressed files.

bzegrep, bzfgrep, bzgrep

bzegrep, bzfgrep, and bzgrep invoke either egrep, fgrep, or grep (respectively) on bzip2−compressed files.

bzip2

bzip2 compresses files using the Burrows−Wheeler block sorting text compression algorithm and Huffman coding. Compression is generally considerably better than that achieved by more conventional LZ77/LZ78−based compressors and approaches the performance of the PPM family of statistical compressors.

bzip2recover

bzip2recover recovers data from damaged bzip2 files.

bzless

bzless is a filter which allows examination of compressed or plain text files, one screenful at a time on a soft−copy terminal, like less.

bzmore

bzmore is a filter which allows examination of compressed or plain text files, one screenful at a time on a soft−copy terminal, like more.

Library Files

libbz2.a, libbz2.so (link to libbz2.so.1.0), libbz2.so.1.0 (link to libbz2.so.1.0.2) and libbz2.so.1.0.2

libbz2

libbz2 is the library for implementing lossless, block−sorting data compression, using the Burrows−Wheeler algorithm.

Bzip2 Installation Dependencies

Last checked against version 1.0.1.

Bash: sh Binutils: ar, as, ld, ranlib Fileutils: cp, ln, rm Gcc: cc1, collect2, cpp0, gcc Make: make

Installing Diffutils–2.8.1

```
Estimated build time:           0.39 SBU
Estimated required disk space:  10 MB
```

Installation of Diffutils

Install Diffutils by running the following commands:

```
LDFLAGS=-static CPPFLAGS=-Dre_max_failures=re_max_failures2 \
    ./configure --prefix=$LFS/static --disable-nls &&
make &&
make install
```

Command explanations

CPPFLAGS=-Dre_max_failures=re_max_failures2: The CPPFLAGS variable is a variable that's read by the cpp program (C PreProcessor). The value of this variable tells the preprocessor to replace every instance of re_max_failures it finds by re_max_failures2 before handing the source file to the compiler itself for compilation. This package has problems linking statically on systems that run an older Glibc version and this construction fixes that problem.

Contents of Diffutils

Last checked against version 2.8.1.

Program Files

cmp, diff, diff3 and sdiff

Descriptions

cmp and diff

cmp and diff both compare two files and report their differences. Both programs have extra options which compare files in different situations.

diff3

The difference between diff and diff3 is that diff compares 2 files, diff3 compares 3 files.

sdiff

sdiff merges two files and interactively outputs the results.

Diffutils Installation Dependencies

Last checked against version 2.7.

Bash: sh Binutils: ld, as Diffutils: cmp Fileutils: chmod, cp, install, mv, rm
Gcc: cc1, collect2, cpp0, gcc Grep: egrep, grep Make: make Sed: sed Sh−utils: date, hostname
Textutils: cat, tr

Installing Fileutils−4.1

```
Estimated build time:           0.94 SBU
Estimated required disk space:  40 MB
```

Installation of Fileutils

The programs from a statically linked Fileutils package may cause segmentation faults on certain systems, if your distribution has Glibc−2.2.3 installed. It seems to happen mostly on machines powered by an AMD CPU, but there is a case or two where an Intel system is affected as well. If your system falls in this category, apply the patch.

Note that in some cases using this patch will result in not being able to compile this package at all, even when your system has an AMD CPU and has Glibc−2.2.3 (or higher) installed. If that's the case, you'll need to remove the fileutils−4.1 directory and unpack it again from the tarball before continuing. We believe this may be the case when your distribution has altered Glibc−2.2.3 somehow, but details are unavailable at this time.

To fix this package to compile properly on AMD/Glibc−2.2.3 machines, run the following command. Do *not* attempt this fix if you don't have Glibc−2.2.3 installed. It will more than likely result in all kinds of compile time problems.

```
patch −Np1 −i ../fileutils−4.1.patch
```

Install Fileutils by running the following commands:

```
LDFLAGS=-static \
    ./configure --disable-nls --prefix=$LFS/static &&
make &&
make install
```

Once you have installed Fileutils, you can test whether the segmentation fault problem has been avoided by running **`$LFS/static/bin/ls`**. If this works, then you are OK. If not, then you need to re−do the installation with the patch if you didn't use it, or without the patch if you did use it.

Command explanations

`patch −Np1 −i ../fileutils−4.1.patch:` This is used to fix a problem with building fileutils statically on glibc 2.2.3 systems. If this isn't done, then there is the possibility of all of the fileutils programs causing segmentation faults once chroot is entered in Chapter 6.

Contents of Fileutils

Last checked against version 4.1.

Program Files

chgrp, chmod, chown, cp, dd, df, dir, dircolors, du, install, ln, ls, mkdir, mkfifo, mknod, mv, rm, rmdir, shred, sync, touch and vdir

Descriptions

chgrp

chgrp changes the group ownership of each given file to the named group, which can be either a group name or a numeric group ID.

chmod

chmod changes the permissions of each given file according to mode, which can be either a symbolic representation of changes to make or an octal number representing the bit pattern for the new permissions.

chown

chown changes the user and/or group ownership of each given file.

cp

cp copies files from one place to another.

dd

dd copies a file (from the standard input to the standard output, by default) with a user−selectable blocksize, while optionally performing conversions on it.

df

df displays the amount of disk space available on the filesystem containing each file name argument. If no file name is given, the space available on all currently mounted filesystems is shown.

dir, ls and vdir

dir and vdir are versions of ls with different default output formats. These programs list each given file or directory name. Directory contents are sorted alphabetically. For ls, files are, by default, listed in columns sorted vertically if the standard output is a terminal; otherwise they are listed one per line. For dir, files are, by default, listed in columns sorted vertically. For vdir, files are, by default, listed in long format.

dircolors

dircolors outputs commands to set the LS_COLOR environment variable. The LS_COLOR variable is use to change the default color scheme used by ls and related utilities.

du

du displays the amount of disk space used by each file or directory listed on the command−line and by each of their subdirectories.

install

install copies files and sets their permission modes and, if possible, their owner and group.

ln

ln makes hard or soft (symbolic) links between files.

mkdir

mkdir creates directories with a given name.

mkfifo

mkfifo creates a FIFO with each given name.

mknod

mknod creates a FIFO, character special file or block special file with the given file name.

mv

mv moves files from one directory to another or renames files, depending on the arguments given to mv.

rm

rm removes files or directories.

rmdir

rmdir removes directories, if they are empty.

shred

shred deletes a file securely, overwriting it first so that its contents can't be recovered.

sync

sync forces changed blocks to disk and updates the super block.

touch

touch changes the access and modification times of each given file to the current time. Files that do not exist are created empty.

Fileutils Installation Dependencies

Last checked against version 4.1.

Bash: sh Binutils: ar, as, ld, ranlib Diffutils: cmp
Fileutils: chmod, cp, install, ln, ls, mkdir, mv, rm, rmdir Gettext: msgfmt, xgettext
Gcc: cc, cc1, collect2, cpp0, gcc Grep: egrep, fgrep, grep Make: make Perl: perl Sed: sed
Sh−utils: basename, echo, expr, hostname, sleep, uname Texinfo: install−info Textutils: cat, tr

Installing Findutils−4.1

```
Estimated build time:           0.12 SBU
Estimated required disk space:  8 MB
```

Installing Findutils

This package requires its patch to be applied before you can install it. Make sure it's unpacked before running the installation commands.

Install Findutils by running the following commands:

```
patch −Np1 −i ../findutils−4.1.patch &&
CPPFLAGS=−Dre_max_failures=re_max_failures2 \
    ./configure −−prefix=$LFS/static &&
make LDFLAGS=−static &&
make install
```

Command explanations

`patch −Np1 −i ../findutils−4.1.patch`: This patch is to fix some compilation errors by avoiding a variable conflict and changing some bad syntax.

Contents of Findutils

Last checked against version 4.1.

Program Files

bigram, code, find, frcode, locate, updatedb and xargs

Descriptions

bigram

bigram is used together with code to produce older−style locate databases. To learn more about these last three programs, read the locatedb.5 manual page.

code

code is the ancestor of frcode. It was used in older−style locate databases.

find

The find program searches for files in a directory hierarchy which match a certain criteria. If no criteria is given, it lists all files in the current directory and its subdirectories.

frcode

frcode is called by updatedb to compress the list of file names using front−compression, which reduces the database size by a factor of 4 to 5.

locate

locate scans a database which contains all files and directories on a filesystem. This program lists the files and directories in this database matching a certain criteria. If a user is looking for a file this program will scan the database and tell him exactly where the files he requested are located. This only makes sense if the locate database is fairly up−to−date, else it will provide out−of−date information.

updatedb

The updatedb program updates the locate database. It scans the entire file system (including other file systems that are currently mounted unless it is told not to do so) and puts every directory and file it finds into the database that's used by the locate program, which retrieves this information. It's good practice to update this database once a day to have it up−to−date whenever it is needed.

xargs

The xargs command applies a command to a list of files. If there is a need to perform the same command on multiple files, a list can be created that names all those files (one per line) and xargs can perform that command on those files.

Findutils Installation Dependencies

Last checked against version 4.1.

Bash: sh Binutils: ar, as, ld, ranlib Diffutils: cmp Fileutils: chmod, cp, install, mv, rm
Grep: egrep, grep Gcc: cc1, collect2, cpp0, gcc Make: make Patch: patch Sed: sed
Sh−utils: basename, date, echo, hostname Textutils: cat, tr

Installing Gawk–3.1.1

```
Estimated build time:           0.39 SBU
Estimated required disk space:  17 MB
```

Installation of Gawk

Install Gawk by running the following commands:

```
CPPFLAGS=-Dre_max_failures=re_max_failures2 \
    ./configure --prefix=$LFS/static --disable-nls &&
make LDFLAGS=-static &&
make install
```

Contents of Gawk

Last checked against version 3.1.1.

Program Files

awk, gawk, gawk–3.1.1, grcat, igawk, pgawk, pgawk–3.1.1, pwcat

Descriptions

awk

awk is a symbolic link to gawk.

gawk, gawk–3.1.1

gawk is the GNU implementation of awk, a pattern scanning and processing language.

grcat

grcat concatenates the group database, /etc/group.

igawk

igawk is a shell script which gives gawk the ability to include files.

pgawk, pgawk–3.1.1

pgawk is the profiling version of gawk.

pwcat

pwcat concatenates the password database, /etc/passwd.

Gawk Installation Dependencies

Last checked against version 3.1.0.

(No dependencies checked yet.)

Installing GCC−3.2

```
Estimated build time:           9.48 SBU
Estimated required disk space:  326 MB
```

Installation of GCC

This package requires its patch to be applied before you can install it. Make sure it's unpacked before running the installation commands.

This package is known to behave badly when you have changed its default optimization flags (including the −march and −mcpu options). GCC is best left alone. Therefore, if you have defined any environment variables that override default optimizations, such as CFLAGS and CXXFLAGS, we recommend unsetting or modifying them when building GCC. You have been warned.

Install GCC by running the following commands:

```
patch −Np1 −i ../gcc−3.2.patch &&
patch −Np1 −i ../gcc−3.2−nofixincludes.patch &&
mkdir ../gcc−build &&
cd ../gcc−build &&
../gcc−3.2/configure −−prefix=/static −−enable−languages=c \
   −−disable−nls −−disable−shared &&
echo "#define HAVE_GAS_HIDDEN 1" >> gcc/auto−host.h &&
make BOOT_LDFLAGS=−static bootstrap &&
make prefix=$LFS/static install &&
ln −s gcc $LFS/static/bin/cc
```

Command explanations

`patch −Np1 −i ../gcc−3.2.patch`: This patch fixes a few bugs. In particular it contains the "copy fix" and "var fix" documented at http://www.zipworld.com.au/~gschafer/lfs−tweaks.html.

`patch −Np1 −i ../gcc−3.2−nofixincludes.patch`: This patch prevents the fixincludes script from running.

`−−prefix=/static`: This is NOT a typo. GCC hard codes some paths while compiling and so we need to pass /static as the prefix during ./configure. We pass the real install prefix during the make install command later.

`−−enable−languages=c`: This builds the C compiler. The C++ compiler will be built in Chapter 6, when we rebuild GCC. Other compilers are available as well. If they are needed, the −−enable−languages parameter may be omitted.

`echo "#define HAVE_GAS_HIDDEN 1"`: This defines the .hidden assembler directive so that we don't build a faulty Glibc later on.

`make BOOT_LDFLAGS=−static`: This is the equivalent to make LDFLAGS=−static as we use with other packages to compile them statically.

`ln −s gcc $LFS/static/bin/cc`: This creates the $LFS/static/bin/gcc symlink, which some packages need.

Contents of GCC

Last checked against version 3.1.

Program Files

c++, c++filt, cc (link to gcc), cc1, cc1plus, collect2, cpp, cpp0, g++, gcc, gccbug, gcov and tradcpp0

Descriptions

cc, cc1, cc1plus, gcc

These are the C compiler. A compiler translates source code in text format to a format that a computer understands. After a source code file is compiled into an object file, a linker will create an executable file from one or more of these compiler generated object files.

c++, cc1plus, g++

These are the C++ compiler, the equivalent of cc and gcc etc.

c++filt

The C++ language provides function overloading, which means that it is possible to write many functions with the same name (providing each takes parameters of different types). All C++ function names are encoded into a low−level assembly label (this process is known as mangling). The c++filt program does the inverse mapping: it decodes (demangles) low−level names into user−level names so that the linker can keep these overloaded functions from clashing.

collect2

collect2 assists with the compilation of constructors.

cpp, cpp0

cpp pre−processes a source file, such as including the contents of header files into the source file. Simply add a line, such as #include <filename>, to your source file. The preprocessor will insert the contents of the included file into the source file.

gccbug

gccbug is a shell script which is used to simplify the creation of bug reports.

gcov

gcov analyzes programs to help create more efficient, faster running code through optimization.

tradcpp0

No description is currently available.

Library Files

libgcc.a, libgcc_eh.a, libgcc_s.so, libiberty.a, libstdc++.[a,so], libsupc++.a

Descriptions

libgcc, libgcc_eh, libgcc_s

Run−time support files for gcc.

libiberty

libiberty is a collection of subroutines used by various GNU programs including getopt, obstack, strerror, strtol and strtoul.

libstdc++

libstdc++ is the C++ library. It is used by C++ programs and contains functions that are frequently used in C++ programs. This way the programmer doesn't have to write certain functions (such as writing a string of text to the screen) from scratch every time he creates a program.

libsupc++

libsupc++ provides support for the c++ programming language. Among other things, libsupc++ contains routines for exception handling.

GCC Installation Dependencies

Last checked against version 2.95.3.

Bash: sh Binutils: ar, as, ld, nm, ranlib Diffutils: cmp
Fileutils: chmod, cp, ln, ls, mkdir, mv, rm, touch Find: find Gcc: cc, cc1, collect2, cpp0, gcc
Grep: egrep, grep Make: make Patch: patch Sed: sed
Sh−utils: basename, dirname, echo, expr, hostname, sleep, true, uname Tar: tar
Texinfo: install−info, makeinfo Textutils: cat, tail, tr

Installing Grep–2.5

```
Estimated build time:           0.26 SBU
Estimated required disk space:  5 MB
```

Installation of Grep

Install Grep by running the following commands:

```
LDFLAGS=-static CPPFLAGS=-Dre_max_failures=re_max_failures2 \
    ./configure --prefix=$LFS/static --disable-nls \
    --disable-perl-regexp &&
make &&
make install
```

Command explanations

`--disable-perl-regexp:` This configure option makes sure Grep is not linked against the PCRE library, which is often only available as a shared library in distributions. Not using this option might result in a compilation error.

Contents of Grep

Last checked against version 2.5.

Program Files

egrep (link to grep), fgrep (link to grep) and grep

Descriptions

egrep

egrep prints lines from files matching an extended regular expression pattern.

fgrep

fgrep prints lines from files matching a list of fixed strings, separated by newlines, any of which is to be matched.

grep

grep prints lines from files matching a basic regular expression pattern.

Grep Installation Dependencies

Last checked against version 2.4.2.

Autoconf: autoconf, autoheader Automake: aclocal, automake Bash: sh Binutils: as, ld Diffutils: cmp Fileutils: chmod, install, ls, mkdir, mv, rm Gettext: msgfmt, xgettext

Gcc: cc, cc1, collect2, cpp0, gcc Glibc: getconf Grep: egrep, fgrep, grep M4: m4 Make: make
Gawk: gawk Sed: sed Sh−utils: basename, echo, expr, hostname, sleep, uname
Texinfo: install−info, makeinfo Textutils: cat, tr

Installing Gzip−1.2.4a

```
Estimated build time:           0.04 SBU
Estimated required disk space:  2 MB
```

Installation of Gzip

Install Gzip by running the following commands:

```
./configure --prefix=$LFS/static &&
make LDFLAGS=-static &&
make install
```

Contents of Gzip

Last checked against version 1.2.4a.

Program Files

gunzip (link to gzip), gzexe, gzip, uncompress (link to gunzip), zcat (link to gzip), zcmp, zdiff, zforce, zgrep, zmore and znew

Description

gunzip, uncompress

gunzip and uncompress decompress files which are compressed with gzip.

gzexe

gzexe allows you to compress executables in place and have them automatically uncompress and execute when they are run (at a penalty in performance).

gzip

gzip reduces the size of the named files using Lempel−Ziv coding (LZ77).

zcat

zcat uncompresses, and writes to standard output, either a list of files on the command line or a file being read from standard input.

zcmp

zcmp invokes the cmp program on compressed files.

zdiff

zdiff invokes the diff program on compressed files.

zforce

zforce forces a .gz extension on all gzip files so that gzip will not compress them twice. This can be useful for files with names truncated after a file transfer.

zgrep

zgrep invokes the grep program on compressed files.

zmore

zmore is a filter which allows examination of compressed or plain text files, one screen at a time on a soft−copy terminal (similar to the more program).

znew

znew re−compresses files from .Z (compress) format to .gz (gzip) format.

Gzip Installation Dependencies

Last checked against version 1.2.4a.

Bash: sh Binutils: as, ld, nm Fileutils: chmod, cp, install, ln, mv, rm
Gcc: cc1, collect2, cpp, cpp0, gcc Grep: egrep, grep Make: make Sed: sed Sh−utils: hostname
Textutils: cat, tr

Installing Make−3.79.1

```
Estimated build time:           0.26 SBU
Estimated required disk space:  8 MB
```

Installation of Make

Install Make by running the following commands:

```
./configure --prefix=$LFS/static --disable-nls &&
make LDFLAGS=-static &&
make install
```

During the make install phase you will see this warning:

```
chgrp: changing group of `/mnt/lfs/static/bin/make': Operation not permitted
/mnt/lfs/static/bin/make needs to be owned by group kmem and setgid;
otherwise the `-l' option will probably not work. You may need special
privileges to complete the installation of /mnt/lfs/static/bin/make.
```

You can safely ignore this warning. The make program doesn't actually need to be owned by group kmem and setgid for the *−l* option to work. (This option tells make not to start any new jobs when a certain load on the system has been reached.)

Contents of Make

Last checked against version 3.79.1.

Program files

make

Descriptions

make

make determines automatically which pieces of a large program need to be recompiled, and issues the commands to recompile them.

Make Installation Dependencies

Last checked against version 3.79.1.

Autoconf: autoconf, autoheader Automake: aclocal, automake Bash: sh Binutils: as, ld Diffutils: cmp
Fileutils: chgrp, chmod, install, ls, mv, rm Gcc: cc, cc1, collect2, cpp0, gcc Glibc: getconf
Grep: egrep, fgrep, grep M4: m4 Make: make Gawk: gawk Sed: sed
Sh−utils: basename, echo, expr, hostname, sleep, uname Texinfo: install−info, makeinfo
Textutils: cat, tr

Installing Patch−2.5.4

```
Estimated build time:           0.10 SBU
Estimated required disk space:  3 MB
```

Installation of Patch

Install Patch by running the following commands:

```
CPPFLAGS=-D_GNU_SOURCE \
    ./configure --prefix=$LFS/static &&
make LDFLAGS=-static &&
make install
```

Command explanations

`CPPFLAGS=-D_GNU_SOURCE:` This flag fixes installation problems of this package on PPC and m68k platforms (that we know of). It doesn't hurt compilation on other platforms, such as x86, so we do it by default.

Contents of Patch

Last checked against version 2.5.4.

Program Files

patch

Descriptions

patch

The patch program modifies a file according to a patch file. A patch file usually is a list created by the diff program that contains instructions on how an original file needs to be modified. Patch is used a lot for source code patches since it saves time and space. Imagine a package that is 1 MB in size. The next version of that package only has changes in two files of the first version. It can be shipped as an entirely new package of 1 MB or just as a patch file of 1 KB which will update the first version to make it identical to the second version. So if the first version was downloaded already, a patch file avoids a second large download.

Patch Installation Dependencies

Last checked against version 2.5.4.

Bash: sh Binutils: as, ld Diffutils: cmp Fileutils: chmod, install, mv, rm
Gcc: cc, cc1, collect2, cpp0, gcc Glibc: getconf Grep: egrep, grep Make: make Sed: sed
Sh−utils: echo, expr, hostname, uname Textutils: cat, tr

Installing Sed−3.02

```
Estimated build time:           0.09 SBU
Estimated required disk space:  2 MB
```

Installation of Sed

Install Sed by running the following commands:

```
CPPFLAGS=-Dre_max_failures=re_max_failures2 \
    ./configure --prefix=$LFS/static &&
make LDFLAGS=-static &&
make install
```

Contents of Sed

Last checked against version 3.02.

Program Files

sed

Descriptions

sed

sed is a stream editor. A stream editor is used to perform basic text transformations on an input stream (a file or input from a pipeline).

Sed Installation Dependencies

Last checked against version 3.02.

Autoconf: autoconf, autoheader Automake: aclocal, automake Bash: sh Binutils: ar, as, ld, ranlib
Diffutils: cmp Fileutils: chmod, install, ls, mv, rm Gcc: cc1, collect2, cpp0, gcc Glibc: getconf
Grep: egrep, fgrep, grep M4: m4 Make: make Gawk: gawk Sed: sed
Sh−utils: echo, expr, hostname, sleep Texinfo: install−info, makeinfo Textutils: cat, tr

Installing Sh–utils–2.0

```
Estimated build time:           0.47 SBU
Estimated required disk space:  42 MB
```

Installation of Sh–utils

Before Sh–utils is installed, the sh–utils patch file may need to be applied. This patch is needed to avoid a conflict of variable names with certain Glibc versions (usually glibc–2.1.x) when compiling sh–utils statically. It is however safe to apply the patch even if you are running a different glibc version. So, if you aren't sure, it's best to apply it.

```
patch -Np1 -i ../sh-utils-2.0.patch
```

Install Sh–utils by running the following commands:

```
./configure --prefix=$LFS/static \
    --disable-nls &&
make LDFLAGS=-static &&
make install
```

During the make install stage you will see the following warning:

```
WARNING: insufficient access; not installing su
NOTE: to install su, run 'make install-root' as root
```

You can safely ignore that warning. You need to be logged in as root in order to install su the way Sh–utils wants to install it, which is being suid root. Because we don't need su during Chapter 6, and su will be properly installed when we re–install Sh–utils in Chapter 6, you can just pretend you didn't see it.

Contents of Sh–utils

Last checked against version 2.0.

Program Files

basename, chroot, date, dirname, echo, env, expr, factor, false, groups, hostid, id, logname, nice, nohup, pathchk, pinky, printenv, printf, pwd, seq, sleep, stty, su, tee, test, true, tty, uname, uptime, users, who, whoami and yes

Descriptions

basename

basename strips directory and suffixes from filenames.

chroot

chroot runs a command or interactive shell with special root directory.

date

date displays the current time in a specified format, or sets the system date.

dirname

dirname strips non−directory suffixes from file name.

echo

echo displays a line of text.

env

env runs a program in a modified environment.

expr

expr evaluates expressions.

factor

factor prints the prime factors of all specified integer numbers.

false

false always exits with a status code indicating failure.

groups

groups prints a user's group memberships.

hostid

hostid prints the numeric identifier (in hexadecimal) for the current host.

id

id prints the effective user and group IDs of the current user or a given user.

logname

logname prints the current user's login name.

nice

nice runs a program with modified scheduling priority.

nohup

nohup runs a command immune to hangups, with output to a log file.

pathchk

pathchk checks whether file names are valid or portable.

pinky

pinky is a lightweight finger utility which retrieves information about a certain user.

printenv

printenv prints all or part of the environment.

printf

printf formats and prints data (the same as the C printf function).

pwd

pwd prints the name of the current/working directory.

seq

seq prints numbers in a certain range with a certain increment.

sleep

sleep delays for a specified amount of time.

stty

stty changes and prints terminal line settings.

su

su runs a shell with substitute user and group IDs.

tee

tee reads from standard input and writes to standard output and files.

test

test checks file types and compares values.

true

true always exits with a status code indicating success.

tty

tty prints the file name of the terminal connected to standard input.

uname

uname prints system information.

uptime

uptime tells how long the system has been running.

users

users prints the user names of users currently logged in to the current host.

who

who shows who is logged on.

whoami

whoami prints the user name associated with the current effective user ID.

yes

yes outputs 'y' or a given string repeatedly, until killed.

Sh−utils Installation Dependencies

Last checked against version 2.0.

Autoconf: autoconf, autoheader Automake: aclocal, automake Bash: sh Binutils: ar, as, ld, ranlib Diffutils: cmp Fileutils: chmod, chown, install, ls, mv, rm Gettext: msgfmt, xgettext Gcc: cc, cc1, collect2, cpp0, gcc Glibc: getconf Grep: egrep, fgrep, grep M4: m4 Make: make Gawk: gawk Perl: perl Sed: sed Sh−utils: basename, echo, expr, hostname, sleep, uname Tar: tar Texinfo: install−info, makeinfo Textutils: cat, tr

Installing Tar−1.13

```
Estimated build time:           0.25 SBU
Estimated required disk space:  10 MB
```

Installation of Tar

To be able to directly use bzip2 files with tar, use the tar patch available from the LFS FTP site. This patch will add the −j option to tar which works the same as the −z option to tar (which can be used for gzip files).

Apply the patch by running the following command:

```
patch −Np1 −i ../tar−1.13.patch
```

Install Tar by running the following commands:

```
./configure −−prefix=$LFS/static −−disable−nls &&
make LDFLAGS=−static &&
make install
```

Contents of Tar

Last checked against version 1.13.

Program Files

rmt and tar

Descriptions

rmt

rmt is a program used by the remote dump and restore programs in manipulating a magnetic tape drive through an interprocess communication connection.

tar

tar is an archiving program designed to store and extract files from an archive file known as a tar file.

Tar Installation Dependencies

Last checked against version 1.13.

Autoconf: autoconf, autoheader Automake: aclocal, automake Bash: sh Binutils: ar, as, ld, ranlib
Diffutils: cmp Fileutils: chmod, install, ls, mv, rm Gettext: msgfmt, xgettext
Gcc: cc, cc1, collect2, cpp0, gcc Glibc: getconf Grep: egrep, fgrep, grep M4: m4 Make: make
Gawk: gawk Net−tools: hostname Patch: patch Sed: sed Sh−utils: basename, echo, expr, sleep, uname
Texinfo: install−info, makeinfo Textutils: cat, tr

Installing Texinfo−4.2

```
Estimated build time:           0.47 SBU
Estimated required disk space:  19 MB
```

Installation of Texinfo

Install Texinfo by running the following commands:

```
LDFLAGS=-static ./configure --prefix=$LFS/static \
    --disable-nls &&
make &&
make install
```

Contents of Texinfo

Last checked against version 4.2.

Program Files

info, infokey, install−info, makeinfo, texi2dvi and texindex

Descriptions

info

The info program reads Info documents, usually contained in the /usr/share/info directory. Info documents are like man(ual) pages, but they tend to go deeper than just explaining the options to a program.

infokey

infokey compiles a source file containing Info customizations into a binary format.

install−info

The install−info program updates the info entries. When the info program is run, a list with available topics (ie: available info documents) will be presented. The install−info program is used to maintain this list of available topics. If info files are removed manually, you must also delete the topic in the index file. This program is used for that. It also works the other way around when info documents are added.

makeinfo

The makeinfo program translates Texinfo source documents into various formats. Available formats are: info files, plain text and HTML.

texi2dvi

The texi2dvi program prints Texinfo documents.

texindex

The texindex program is used to sort Texinfo index files.

Texinfo Installation Dependencies

Last checked against version 4.0.

Bash: sh Binutils: ar, as, ld, ranlib Diffutils: cmp Fileutils: chmod, install, ln, ls, mkdir, mv, rm
Gcc: cc1, collect2, cpp0, gcc Grep: egrep, fgrep, grep Make: make Sed: sed
Sh−utils: basename, echo, expr, hostname, sleep Texinfo: makeinfo Textutils: cat, tr

Installing Textutils−2.1

```
Estimated build time:           0.95 SBU
Estimated required disk space:  49 MB
```

Installation of Textutils

Install Textutils by running the following commands:

```
CPPFLAGS=-Dre_max_failures=re_max_failures2 \
    ./configure --prefix=$LFS/static \
    --disable-nls &&
make LDFLAGS=-static &&
make install
```

Contents of Textutils

Last checked against version 2.0.

Program Files

cat, cksum, comm, csplit, cut, expand, fmt, fold, head, join, md5sum, nl, od, paste, pr, ptx, sort, split, sum, tac, tail, tr, tsort, unexpand, uniq and wc

Descriptions

cat

cat concatenates file(s) or standard input to standard output.

cksum

cksum prints CRC checksum and byte counts of each specified file.

comm

comm compares two sorted files line by line.

csplit

csplit outputs pieces of a file separated by (a) pattern(s) to files xx01, xx02, ..., and outputs byte counts of each piece to standard output.

cut

cut prints selected parts of lines from specified files to standard output.

expand

expand converts tabs in files to spaces, writing to standard output.

fmt

fmt reformats each paragraph in the specified file(s), writing to standard output.

fold

fold wraps input lines in each specified file (standard input by default), writing to standard output.

head

head prints the first xx (10 by default) lines of each specified file to standard output.

join

join joins lines of two files on a common field.

md5sum

md5sum prints or checks MD5 checksums.

nl

nl writes each specified file to standard output, with line numbers added.

od

od writes an unambiguous representation, octal bytes by default, of a specified file to standard output.

paste

paste writes lines consisting of the sequentially corresponding lines from each specified file, separated by TABs, to standard output.

pr

pr paginates or columnates files for printing.

ptx

ptx produces a permuted index of file contents.

sort

sort writes sorted concatenation of files to standard output.

split

split outputs fixed–size pieces of an input file to PREFIXaa, PREFIXab, ...

sum

sum prints checksum and block counts for each specified file.

tac

tac writes each specified file to standard output, last line first.

tail

tail print the last xx (10 by default) lines of each specified file to standard output.

tr

tr translates, squeezes, and/or deletes characters from standard input, writing to standard output.

tsort

tsort writes totally ordered lists consistent with the partial ordering in specified files.

unexpand

unexpand converts spaces in each file to tabs, writing to standard output.

uniq

uniq removes duplicate lines from a sorted file.

wc

wc prints line, word and byte counts for each specified file and a total line, if more than one file is specified.

Textutils Installation Dependencies

Last checked against version 2.0.

Autoconf: autoconf, autoheader Automake: aclocal, automake Bash: sh Binutils: ar, as, ld, ranlib
Diffutils: cmp Fileutils: chmod, install, ls, mv, rm Gettext: msgfmt, xgettext
Gcc: cc, cc1, collect2, cpp0, gcc Glibc: getconf Grep: egrep, fgrep, grep M4: m4 Make: make
Gawk: gawk Net−tools: hostname Perl: perl Sed: sed Sh−utils: basename, echo, expr, sleep, uname
Tar: tar Texinfo: install−info, makeinfo Textutils: cat, tr

Installing Util−linux−2.11u

```
Estimated build time:           0.09 SBU
Estimated required disk space:  9 MB
```

Installation of Util−linux

We only need the mount and umount programs at the moment, so we won't be compiling the entire package.

Install Util−linux by running the following commands:

```
./configure &&
make -C lib &&
make -C mount LDFLAGS=-static mount umount &&
cp mount/{mount,umount} $LFS/static/bin
```

Util−linux Installation Dependencies

Last checked against version 2.11n.

Bash: sh Binutils: as, ld Diffutils: cmp Fileutils: chgrp, chmod, cp, install, ln, mv, rm
Gettext: msgfmt, xgettext Gcc: cc, cc1, collect2, cpp, cpp0 Glibc: rpcgen Grep: grep Make: make
Sed: sed Sh−utils: uname, whoami Textutils: cat

Chapter 6. Installing basic system software

Introduction

In this chapter we enter the building site, and start constructing our LFS system in earnest. That is, we chroot into our temporary mini Linux system, create some auxiliary things, and then start installing all the packages, one by one.

The installation of all this software is pretty straightforward, and you will probably think it would be much shorter to give here the generic installation instructions and explain in full only the installation of those packages that require an alternate method. Although we agree with that, we nevertheless choose to give the full instructions for each and every package, simply to minimize the possibilities for mistakes.

If you plan to use compiler optimizations in this chapter, take a look at the optimization hint at http://hints.linuxfromscratch.org/hints/optimization.txt. Compiler optimizations can make a program run faster, but they may also cause compilation difficulties. If a package refuses to compile when using optimization, try to compile it without optimization and see if the problem goes away.

The order in which packages are installed in this chapter has to be strictly followed, to ensure that no program gets a path referring to `/static` hard−wired into it. For the same reason, *do not* compile packages in parallel. Compiling in parallel may save you some time (especially on dual−CPU machines), but it could result in a program containing a hard−wired path to `/static`, which will cause the program to stop working when the static directory is removed.

About debugging symbols

Most programs and libraries are, by default, compiled with debugging symbols included (with gcc option −g).

When debugging a program or library that was compiled with debugging information included, the debugger can give you not only memory addresses but also the names of the routines and variables.

But the inclusion of these debugging symbols enlarges a program or library significantly. To get an idea of the amount of space these symbols occupy, have a look at the following:

- 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 a little, depending on which compiler was used and which C library. But when comparing programs with and without debugging symbols, the difference will generally be a factor between 2 and 5.

As most people will probably never use a debugger on their system software, a lot of disk space can be regained by removing these symbols .

To remove debugging symbols from a binary (which must be an a.out or ELF binary), run **`strip --strip-debug filename`**. Wildcards can be used to treat multiple files (use something like **`strip --strip-debug $LFS/static/bin/*`**).

For your convenience, Chapter 9 includes one simple command to strip all debugging symbols from all programs and libraries on your system. Additional information on optimization can be found in the hint at http://hints.linuxfromscratch.org/hints/optimization.txt.

Entering the chroot environment

It is time to enter the chroot environment in order to begin installing the packages we need. Before you can chroot, however, you need to become *root*, since only *root* can execute the **`chroot`** command.

Become *root* and run the following command to enter the chroot environment:

```
chroot $LFS /static/bin/env -i \
    HOME=/root TERM=$TERM PS1='\u:\w\$ ' \
    PATH=/bin:/usr/bin:/sbin:/usr/sbin:/static/bin \
    /static/bin/bash --login
```

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 you need other variables present, such as CFLAGS or CXXFLAGS, this is a good place to set them again.

From this point on there's no need anymore to use the LFS variable, because everything you do will be restricted to the LFS file system –– since what the shell thinks is `/` is actually `/mnt/lfs`.

You have to make sure all the commands in the rest of this chapter and in the following chapters are run from within the chroot environment. If you ever leave this environment for any reason (rebooting for example), you must remember to again enter chroot and mount proc (discussed later) before continuing with the installations.

Note that the bash prompt will say "I have no name!" This is normal, as the Glibc package hasn't been installed yet.

Changing ownership

The first thing we'll do, now that we're *root*, is change the ownership of the files and directories installed in Chapter 5 to root –– because when later we don't delete the `/static` directory and start adding new users, one of these users might end up owning the statically linked programs, which is not a good idea.

Run the following command to make root the owner of all the statically linked programs:

```
chown -R 0:0 /static
```

The command uses "0:0" instead of "root:root", because there is no way to resolve the name "root", as glibc hasn't been installed yet.

Creating directories

Let's now create some structure in our LFS file system. Let's create a directory tree. Issuing the following commands will create a more or less standard tree:

```
mkdir -p /{bin,boot,dev/pts,etc/opt,home,lib,mnt,proc} &&
mkdir -p /{root,sbin,tmp,usr/local,var,opt} &&
for dirname in /usr /usr/local
    do
    mkdir $dirname/{bin,etc,include,lib,sbin,share,src}
    ln -s share/{man,doc,info} $dirname
    mkdir $dirname/share/{dict,doc,info,locale,man}
    mkdir $dirname/share/{nls,misc,terminfo,zoneinfo}
    mkdir $dirname/share/man/man{1,2,3,4,5,6,7,8}
done &&
mkdir /var/{lock,log,mail,run,spool} &&
mkdir -p /var/{tmp,opt,cache,lib/misc,local} &&
mkdir /opt/{bin,doc,include,info} &&
mkdir -p /opt/{lib,man/man{1,2,3,4,5,6,7,8}} &&
ln -s ../var/tmp /usr
```

Directories are, by default, created with permission mode 755, but this isn't desirable for all directories. We will make two changes: one to the home directory of root, and another to the directories for temporary files.

```
chmod 0750 /root &&
chmod 1777 /tmp /var/tmp
```

The first mode change ensures that not just everybody 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 other users' files from them. The latter is prohibited by the so–called "sticky bit" –– the highest bit in the 1777 bit mask.

Now that the directories are created, move the source tarballs that were downloaded in Chapter 3 to some subdirectory under `/usr/src` (you will have to create the desired subdirectory yourself).

FHS compliance note

We have based our directory tree on the FHS standard (available at http://www.pathname.com/fhs/). Besides the above created tree this standard stipulates the existence of `/usr/local/games` and `/usr/share/games`, but we don't much like these for a base system. However, feel free to make your system FHS–compliant. As to the structure of the `/usr/local/share` subdirectory, the FHS isn't precise, so we created here the directories that we think are needed.

Mounting the proc file system

In order for certain programs to function properly, the proc file system must be available within the chroot environment. As a file system can be mounted as many times and in as many places as you like, it's not a problem that the proc file system is already mounted on your host system –– especially so because proc is a virtual file system.

The proc file system is mounted under `/proc` by running the following command:

```
mount proc /proc -t proc
```

You will most likely get some warning messages from the mount command, such as these:

```
warning: can't open /etc/fstab: No such file or directory
not enough memory
```

Ignore these, they're just due to the fact that the system isn't installed completely yet and some files are missing. The mount itself will be successful and that's all we care about at this point.

Creating the mtab symlink

The next thing to do is to create a symlink pointing from `/etc/mtab` to `/proc/mounts`. This is done using the following command:

```
ln -sf /proc/mounts /etc/mtab
```

Creating this symlink avoids problems which can occur if `/` is mounted read–only and the information in `/etc/mtab` is stale (i.e. out of date). By creating the symlink to `/proc/mounts`, we ensure that the information on currently mounted devices is always up–to–date.

Note that using this symlink requires that you have support for the proc filesystem compiled into your kernel. This support is included by default, and should not be removed unless you *really* know what you are doing, as some more things besides the `/etc/mtab` symlink depend on proc being present. In short, make sure you have proc filesystem support in your kernel.

Creating the bash and sh symlinks

Some programs hard−wire paths to programs which don't exist yet. In order to satisfy these programs, we create the symbolic links `/bin/bash` and `/bin/sh`, both pointing to the static `bash` program.

Create the `/bin/bash` and `/bin/sh` symlinks by running the following commands:

```
ln −s /static/bin/bash /bin/bash &&
ln −s bash /bin/sh
```

Creating the passwd and group files

In order for root to be able to login and for the name "root" to be recognized, there need to be relevant entries in the /etc/passwd and /etc/group files.

Create the /etc/passwd file by running the following command:

```
echo "root:x:0:0:root:/root:/bin/bash" > /etc/passwd
```

The actual password for root (the "x" here is just a placeholder) will be set later.

Create the /etc/group file by running the following command:

```
cat > /etc/group << "EOF"
root:x:0:
bin:x:1:
sys:x:2:
kmem:x:3:
tty:x:4:
tape:x:5:
daemon:x:6:
floppy:x:7:
disk:x:8:
lp:x:9:
dialout:x:10:
audio:x:11:
EOF
```

The created groups aren't part of any standard –– they are the groups that the MAKEDEV script in the next section uses. Besides the group "root", the LSB recommends only a group "bin", with a GID of 1, be present. All other group names and GIDs can be chosen freely by the user, as well–written packages don't depend on GID numbers but use the group's name.

Creating devices (Makedev−1.7)

```
Estimated build time:           0.07 SBU
Estimated required disk space:  50 KB
```

Creating devices

Note that unpacking the MAKEDEV−1.7.bz2 file doesn't create a directory for you to cd into, as the file only contains a script.

Prepare for the creation of the device files by running the following commands:

```
cp MAKEDEV-1.7 /dev/MAKEDEV &&
cd /dev &&
chmod 754 MAKEDEV
```

Most people will now want to create devices by running:

```
./MAKEDEV -v generic
```

But if you intend to use devpts, then run this instead:

```
./MAKEDEV -v generic-nopty
```

Note that if you aren't sure, it's best to use the **`./MAKEDEV -v generic`** command as this will ensure you have all the devices you need. But if you are certain you are going to use devpts, the other command skips creating a set of devices you won't need.

`MAKEDEV` will create hda[1−20] to hdh[1−20] and many more of such disk device nodes, but keep in mind that you probably won't be able to use all of these, due to kernel limits on the maximum number of partitions.

Command explanations

`./MAKEDEV -v generic:` This creates a whole bunch of devices. Normally, these are all the devices you will need. But it is possible that some special devices needed for your hardware configuration are missing. Create these with ./MAKEDEV −v <device>. The **`generic-nopty`** option mostly creates the same devices as **`generic`**, but skips those that aren't needed if you are using devpts.

Contents of MAKEDEV

Last checked against version 1.5.

Program Files

MAKEDEV

Descriptions

MAKEDEV

`MAKEDEV` is a script that creates the necessary static device nodes usually residing in the `/dev` directory. Detailed information on device nodes can be found in the Linux kernel source tree in `Documentation/devices.txt`.

MAKEDEV Installation Dependencies

Last checked against version 1.5.

Bash: sh Fileutils: chmod, chown, cp, ln, mknod, mv, rm Grep: grep Sh−utils: expr, id

Installing Linux–2.4.19

```
Estimated build time:           0.02
Estimated required disk space:  142 MB
```

Installation of the kernel headers

We won't be compiling a new kernel yet –– we'll do that when we have finished the installation of all the packages. But as some packages need the kernel header files, we're going to unpack the kernel archive now, set it up, and copy the header files to where they will be found by these packages.

The kernel headers are copied by running the following commands:

```
ln -s /static/bin/pwd /bin/pwd &&
make mrproper &&
make include/linux/version.h &&
make symlinks &&
mkdir /usr/include/asm &&
cp include/asm/* /usr/include/asm &&
cp -R include/asm-generic /usr/include &&
cp -R include/linux /usr/include &&
touch /usr/include/linux/autoconf.h &&
rm /bin/pwd
```

Command explanations

`ln -s /static/bin/pwd /bin/pwd`: In the kernel source, the path to the `pwd` program is hard–wired as `/bin/pwd`, so we create a temporary symlink to account for that. At the end we remove it again.

`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 and that you shouldn't rely on the source tree being clean after untarring.

`make include/linux/version.h` and **`make symlinks`**: This creates the `include/linux/version.h` file and the platform–specific `include/asm` symlink.

`mkdir /usr/include/asm` , **`cp include/asm/* /usr/include/asm`** and **`cp -R include/asm-generic /usr/include`**: These commands copy the platform–specific assembler kernel header files to `/usr/include/asm` and `/usr/include/asm-generic`.

`cp -R include/linux /usr/include`: This command copies the cross–platform kernel header files to `/usr/include`.

`touch /usr/include/linux/autoconf.h`: This creates an empty `autoconf.h` file. As we do not yet configure the kernel, we have to create this file ourselves for those few kernel header files that make use of it, to avoid compilation failures.

Why we copy the kernel headers and don't symlink them

In the past it was common practice to symlink the `/usr/include/{linux,asm}` directories to `/usr/src/linux/include/{linux,asm}`. This was a *bad* practice, as the following extract from a post by Linus Torvalds to the Linux Kernel Mailing List points out:

```
I would suggest that people who compile new kernels should:

 - not have a single symbolic link in sight (except the one that the
   kernel build itself sets up, namely the "linux/include/asm" symlink
   that is only used for the internal kernel compile itself)

And yes, this is what I do. My /usr/src/linux still has the old 2.2.13
header files, even though I haven't run a 2.2.13 kernel in a _loong_
time. But those headers were what glibc was compiled against, so those
headers are what matches the library object files.

And this is actually what has been the suggested environment for at
least the last five years. I don't know why the symlink business keeps
on living on, like a bad zombie. Pretty much every distribution still
has that broken symlink, and people still remember that the linux
sources should go into "/usr/src/linux" even though that hasn't been
true in a _loong_ time.
```

The essential part is where Linus states that the header files should be *the ones which glibc was compiled against*. These are the headers that should be used when you later compile other packages, as they are the ones that match the object–code library files. By copying the headers, we ensure that they remain available if later you upgrade your kernel.

Note, by the way, that it is perfectly all right to have the kernel sources in `/usr/src/linux`, as long as you don't have the `/usr/include/{linux,asm}` symlinks.

Contents of Linux

Last checked against version 2.4.18.

Support Files

The linux kernel and the linux kernel headers

Descriptions

linux kernel

The Linux kernel is at the core of every Linux system. It's what makes Linux tick. When a computer is turned on and boots a Linux system, the very first piece of Linux software that gets loaded is the kernel. The kernel initializes the system's hardware components: serial ports, parallel ports, sound cards, network cards, IDE controllers, SCSI controllers and a lot more. In a nutshell the kernel makes the hardware available so that the software can run.

linux kernel headers

These are the files we copy to `/usr/include/{linux,asm}` in Chapter 5. They should match those which glibc was compiled against and therefore should *not* be replaced when upgrading the kernel. They are essential for compiling many programs.

Linux Installation Dependencies

Last checked against version 2.4.17.

Bash: sh Binutils: ar, as, ld, nm, objcopy Fileutils: cp, ln, mkdir, mv, rm, touch
Findutils: find, xargs Gcc: cc1, collect2, cpp0, gcc Grep: grep Gzip: gzip Make: make Gawk: awk
Modutils: depmod, genksyms Net−tools: dnsdomainname, hostname Sed: sed
Sh−utils: basename, date, expr, pwd, stty, uname, whoami, yes Textutils: cat, md5sum, sort, tail

Installing Man−pages−1.52

```
Estimated build time:           0.01 SBU
Estimated required disk space:  6 MB
```

Installation of Man−pages

Install Man−pages by running the following command:

```
make install
```

Contents of Man−pages

Last checked against version 1.52.

Support Files

various manual pages that don't come with the packages.

Descriptions

manual pages

Examples of provided manual pages are the manual pages describing all the C and C++ functions, a few important /dev/ files and more.

Man−pages Installation Dependencies

Last checked against version 1.47.

Bash: sh Fileutils: install Make: make

Installing Glibc−2.2.5

```
Estimated build time:           14.71 SBU
Estimated required disk space:  369 MB
```

Installation of Glibc

This package requires its patch to be applied before you can install it. Make sure it's unpacked before running the installation commands.

Before starting to install glibc, you must cd into the glibc−2.2.5 directory and unpack glibc−linuxthreads inside the glibc−2.2.5 directory, not in /usr/src as you normally would do.

This package is known to behave badly when you have changed its default optimization flags (including the −march and −mcpu options). Glibc is best left alone. Therefore, if you have defined any environment variables that override default optimizations, such as CFLAGS and CXXFLAGS, we recommend unsetting or modifying them when building Glibc. You have been warned.

Also, don't pass the −−enable−kernel option to the configure script. It's known to cause segmentation faults when other packages like fileutils, make and tar are linked against it.

Basically, compiling Glibc in any other way than the book suggests is putting your system at very high risk.

Install Glibc by running the following commands:

```
patch −Np1 −i ../glibc−2.2.5−2.patch &&
touch /etc/ld.so.conf &&
mkdir ../glibc−build &&
cd ../glibc−build &&
../glibc−2.2.5/configure −−prefix=/usr −−disable−profile \
    −−enable−add−ons −−libexecdir=/usr/bin &&
echo "cross−compiling = no" > configparms &&
make &&
make install &&
make localedata/install−locales &&
exec /static/bin/bash −−login
```

An alternative to running **`make localedata/install−locales`** is to only install those locales which you need or want. This can be achieved using the localedef command. Information on this can be found in the INSTALL file in the glibc−2.2.5 tree. One thing to note is that the **`localedef`** program assumes that the `/usr/lib/locale` directory exists, so you need to create it first.

The Linux Threads man pages are not going to be installed at this point because it requires a working Perl installation. We'll install Perl later on in this chapter, so we'll come back to the Linux Threads man page installation after that.

During the configure stage you will see the following warning:

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

The missing msgfmt (from the gettext package which we will install later in this chapter) won't cause any problems. msgfmt is used to generate the binary translation files that are used to make your system talk in a different language. Because these translation files have already been generated for you, there is no need for msgfmt. You'd only need msgfmt if you change the translation source files (the `*.po` files in the `po` subdirectory) which would require you to re−generate the binary files.

Command explanations

`patch −Np1 −i ../glibc−2.2.5−2.patch:` This patch converts all occurrences of `$(PERL)` to `/usr/bin/perl` in the `malloc/Makefile` file. This is done because Glibc can't autodetect the location of perl because perl has yet to be installed. The patch also replaces all occurrences of *root* with *0* in the `login/Makefile` file. This is done because Glibc itself isn't installed yet and therefore username to userid resolving isn't working yet, so a **`chown root file`** will fail, however it'll work fine if you use straight IDs.

The patch also contains a few bug fixes and security fixes. In particular it contains the "errlist", "dns resolver", "xdr_array", "calloc", "thread exit", "udivdi3", "math test", "restrict_arr" and "divbyzero" fixes which are documented at http://www.zipworld.com.au/~gschafer/lfs−tweaks.html.

`touch /etc/ld.so.conf:` One of the final steps of the Glibc installation is running ldconfig to update the dynamic loader cache. If this file doesn't exist, the installation will abort with an error that it can't read the file, so we simply create an empty file (the empty file will have Glibc default to using /lib and /usr/lib which is fine).

`−−disable−profile:` This disables the building of libraries with profiling information. This command may be omitted if you plan to do profiling.

`−−enable−add−ons:` This enables the add−on that we install with Glibc, linuxthreads

`−−libexecdir=/usr/bin:` This will cause the pt_chown program to be installed in the /usr/bin directory.

`echo "cross−compiling = no" > configparms:` We do this because we are only building for our own system. Cross−compiling is used, for instance, to build a package for an Apple Power PC on an Intel system. The reason Glibc thinks we're cross−compiling is that it can't compile a test program to determine this, so it automatically defaults to a cross−compiler. Compiling the test program fails because Glibc hasn't been installed yet.

`exec /bin/bash:`This command will start a new bash shell which will replace the current shell. This is done to get rid of the "I have no name!" message in the command prompt, which was caused by bash's inability to resolve a user ID to a user name (which in turn was caused by the absence of Glibc).

Contents of Glibc

Last checked against version 2.2.5.

Program Files

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

Descriptions

catchsegv

catchsegv can be used to create a stack trace when a program terminates with a segmentation fault.

gencat

gencat generates message catalogues.

getconf

getconf displays the system configuration values for filesystem specific variables.

getent

getent gets entries from an administrative database.

glibcbug

glibcbug creates a bug report about glibc and and mails it to the bug email address.

iconv

iconv performs character set conversion.

iconvconfig

iconvconfig creates fastloading iconv module configuration file.

ldconfig

ldconfig configures the dynamic linker run time bindings.

ldd

ldd prints the shared libraries required by each program or shared library specified on the command line.

lddlibc4

lddlibc4 assists ldd with object files.

locale

locale is a Perl program which tells the compiler to enable (or disable) the use of POSIX locales for built−in operations.

localedef

localedef compiles locale specifications.

mtrace

mtrace prints the multicast path from a source to a receiver (an IP trace query).

nscd

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

nscd_nischeck

nscd_nischeck checks whether or not secure mode is necessary for NIS+ lookup.

pcprofiledump

pcprofiledump dumps information generated by PC profiling.

pt_chown

pt_chown sets the owner, group and access permission of the slave pseudo terminal corresponding to the master pseudo terminal passed on file descriptor `3'. This is the helper program for the `grantpt' function. It is not intended to be run directly from the command line.

rpcgen

rpcgen generates C code to implement the RPC protocol.

rpcinfo

rpcinfo makes an RPC call to an RPC server.

sln

sln symbolically links dest to source. It is statically linked, needing no dynamic linking at all. Thus sln is useful to make symbolic links to dynamic libraries if the dynamic linking system for some reason is nonfunctional.

sprof

sprof reads and displays shared object profiling data.

tzselect

tzselect asks the user for information about the current location and outputs the resulting time zone description to standard output.

xtrace

xtrace traces execution of program by printing the currently executed function.

zdump

zdump is the time zone dumper.

zic

zic is the time zone compiler.

Library Files

ld.so, libBrokenLocale.[a,so], libBrokenLocale_p.a, libSegFault.so, libanl.[a,so], libanl_p.a, libbsd−compat.a, libc.[a,so], libc_nonshared.a, libc_p.a, libcrypt.[a,so], libcrypt_p.a, libdl.[a,so], libdl_p.a, libg.a, libieee.a, libm.[a,so], libm_p.a, libmcheck.a, libmemusage.so, libnsl.a, libnsl_p.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_p.a, libresolv.[a,so], libresolv_p.a, librpcsvc.a, librpcsvc_p.a, librt.[a,so], librt_p.a, libthread_db.so, libutil.[a,so] and libutil_p.a

Descriptions

ld.so

ld.so is the helper program for shared library executables.

libBrokenLocale, libBrokenLocale_p

Used by software, such as Mozilla, to solve broken locales.

libSegFault

libSegFault is a segmentation fault signal handler. It tries to catch segfaults.

libanl, libanl_p

libanl is an asynchronous name lookup library.

libbsd−compat

libbsd−compat provides the portability needed in order to run certain programs in Linux.

libc, libc_nonshared, libc_p

These files constitute the main C library. The C library is a collection of commonly used functions in programs. This way a programmer doesn't need to create his own functions for every single task. The most common things like writing a string to the screen are already present and at the disposal of the programmer.

The C library (actually almost every library) comes in two flavors: a dynamic and a static one. In short, when a program uses a static C library, the code from the C library is copied into the executable file. When a program uses a dynamic library, the executable will not contain the code from the C library, but instead a routine that loads the functions from the library at the time the program is run. This means a significant decrease in the file size of a program. The documentation that comes with the C library describes this in more detail, as it is too complicated to explain here in one or two lines.

libcrypt, libcrypt_p

libcrypt is the cryptography library.

libdl, libdl_p

libdl is the dynamic linking interface library.

libg

libg is a runtime library for g++.

libieee

libieee is the IEEE floating point library.

libm, libm_p

libm is the mathematical library.

libmcheck

libmcheck contains code run at boot.

libmemusage

libmemusage is used by memusage to help collect information about the memory usage of a program.

libnsl, libnsl_p

libnsl is the network services library.

libnss_compat, libnss_dns, libnss_files, libnss_hesiod, libnss_nis, libnss_nisplus

The basic idea is to put the implementation of the different services offered to access the databases in separate modules. This has some advantages:

- contributors can add new services without adding them to GNU C library,
- the modules can be updated separately,
-

the C library image is smaller.

libpcprofile

Code used by the kernel to track CPU time spent in functions, source code lines, and instructions.

libpthread, libpthread_p

The POSIX threads library.

libresolv, libresolv_p

Functions in this library provide for creating, sending, and interpreting packets to the Internet domain name servers.

librpcsvc, librpcsvc_p

Functions in this library provide miscellaneous RPC services.

librt, librt_p

Functions in this library provide most of the interfaces specified by the POSIX.1b Realtime Extension.

libthread_db

Functions is this library are useful for building debuggers for multi−threaded programs.

libutil, libutil_p

Contains code for "standard" functions used in many different Unix utilities.

Glibc Installation Dependencies

Last checked against version 2.2.5.

Bash: sh Binutils: ar, as, ld, ranlib, readelf Diffutils: cmp
Fileutils: chmod, cp, install, ln, mknod, mv, mkdir, rm, touch Gcc: cc, cc1, collect2, cpp, gcc
Grep: egrep, grep Gzip: gzip Make: make Gawk: gawk Sed: sed
Sh−utils: date, expr, hostname, pwd, uname Texinfo: install−info, makeinfo
Textutils: cat, cut, sort, tr

Installing GCC−3.2

```
Estimated build time:           13.26 SBU
Estimated required disk space:  221 MB
```

Installation of GCC

This package requires its patch to be applied before you can install it. Make sure it's unpacked before running the installation commands.

This package is known to behave badly when you have changed its default optimization flags (including the −march and −mcpu options). GCC is best left alone. Therefore, if you have defined any environment variables that override default optimizations, such as CFLAGS and CXXFLAGS, we recommend unsetting or modifying them when building Gcc. You have been warned.

Install GCC by running the following commands. These commands will build the C and C++ compiler. Other compilers are available within the gcc package. If you want to build all the other available compilers too, leave out the −−enable−languages=c,c++ option in the configure command. See the GCC documentation for more details on which additional compilers are available.

Note: the build of other compilers is not tested by the people who actively work on LFS.

```
patch -Np1 -i ../gcc-3.2.patch &&
mkdir ../gcc-build &&
cd ../gcc-build &&
../gcc-3.2/configure --prefix=/usr --enable-shared \
    --enable-languages=c,c++ --enable-threads=posix \
    --with-slibdir=/lib --enable-__cxa_atexit \
    --enable-clocale=gnu &&
make bootstrap &&
make install &&
ln -s ../usr/bin/cpp /lib &&
ln -s ../bin/cpp /usr/lib &&
ln -s gcc /usr/bin/cc
```

Command explanations

`--enable-threads=posix:` This enables C++ exception handling for multithreaded code.

`--enable-__cxa_atexit:` This option will result in C++ shared libraries and C++ programs that are interoperable with other linux distributions.

`--enable-clocale=gnu:` There is a risk that some people will build ABI incompatible C++ libraries if they didn't install all of the glibc localedata. Using −−enable−clocale=gnu ensures that the "right thing" is done in all cases. If you don't wish to use this option, then at least build the *de_DE* locale. When GCC finds this specific locale, then the correct locale mode (*gnu*) is implemented.

Contents of GCC

Last checked against version 3.1.

Program Files

c++, c++filt, cc (link to gcc), cc1, cc1plus, collect2, cpp, cpp0, g++, gcc, gccbug, gcov and tradcpp0

Descriptions

cc, cc1, cc1plus, gcc

These are the C compiler. A compiler translates source code in text format to a format that a computer understands. After a source code file is compiled into an object file, a linker will create an executable file from one or more of these compiler generated object files.

c++, cc1plus, g++

These are the C++ compiler, the equivalent of cc and gcc etc.

c++filt

The C++ language provides function overloading, which means that it is possible to write many functions with the same name (providing each takes parameters of different types). All C++ function names are encoded into a low−level assembly label (this process is known as mangling). The c++filt program does the inverse mapping: it decodes (demangles) low−level names into user−level names so that the linker can keep these overloaded functions from clashing.

collect2

collect2 assists with the compilation of constructors.

cpp, cpp0

cpp pre−processes a source file, such as including the contents of header files into the source file. Simply add a line, such as #include <filename>, to your source file. The preprocessor will insert the contents of the included file into the source file.

gccbug

gccbug is a shell script which is used to simplify the creation of bug reports.

gcov

gcov analyzes programs to help create more efficient, faster running code through optimization.

tradcpp0

No description is currently available.

Library Files

libgcc.a, libgcc_eh.a, libgcc_s.so, libiberty.a, libstdc++.[a,so], libsupc++.a

Descriptions

libgcc, libgcc_eh, libgcc_s

Run−time support files for gcc.

libiberty

libiberty is a collection of subroutines used by various GNU programs including getopt, obstack, strerror, strtol and strtoul.

libstdc++

libstdc++ is the C++ library. It is used by C++ programs and contains functions that are frequently used in C++ programs. This way the programmer doesn't have to write certain functions (such as writing a string of text to the screen) from scratch every time he creates a program.

libsupc++

libsupc++ provides support for the c++ programming language. Among other things, libsupc++ contains routines for exception handling.

GCC Installation Dependencies

Last checked against version 2.95.3.

Bash: sh Binutils: ar, as, ld, nm, ranlib Diffutils: cmp
Fileutils: chmod, cp, ln, ls, mkdir, mv, rm, touch Find: find Gcc: cc, cc1, collect2, cpp0, gcc
Grep: egrep, grep Make: make Patch: patch Sed: sed
Sh−utils: basename, dirname, echo, expr, hostname, sleep, true, uname Tar: tar
Texinfo: install−info, makeinfo Textutils: cat, tail, tr

Installing Zlib−1.1.4

```
Estimated build time:           0.07 SBU
Estimated required disk space:  1 MB
```

Installation of Zlib

Install Zlib by running the following commands:

```
./configure --prefix=/usr --shared &&
make LIBS="libz.so.1.1.4 libz.a" &&
make LIBS="libz.so.1.1.4 libz.a" install &&
mv /usr/lib/libz.so.* /lib &&
ln -sf ../../lib/libz.so.1 /usr/lib/libz.so &&
cp zlib.3 /usr/share/man/man3
```

Contents of Zlib

Last checked against version 1.1.4.

Library Files

libz[a,so]

Descriptions

libz

This is the zlib library, which is used by many programs for its compression and uncompression functions.

Zlib Installation Dependencies

No dependencies checked yet.

Installing Findutils−4.1

```
Estimated build time:           0.10 SBU
Estimated required disk space:  3 MB
```

Installing Findutils

This package requires its patch to be applied before you can install it. Make sure it's unpacked before running the installation commands.

Install Findutils by running the following commands:

```
patch -Np1 -i ../findutils-4.1.patch &&
./configure --prefix=/usr &&
make libexecdir=/usr/bin &&
make libexecdir=/usr/bin install
```

FHS compliance notes

By default, the location of the updatedb database is in /usr/var. If you would rather be FHS compliant, you may wish to use another location. The following commands use the database file `/var/lib/misc/locatedb` which is FHS compliant.

```
patch -Np1 -i ../findutils-4.1.patch &&
./configure --prefix=/usr &&
make localstatedir=/var/lib/misc libexecdir=/usr/bin &&
make localstatedir=/var/lib/misc libexecdir=/usr/bin install
```

Command explanations

`patch -Np1 -i ../findutils-4.1.patch`: This patch is to fix some compilation errors by avoiding a variable conflict and changing some bad syntax.

Contents of Findutils

Last checked against version 4.1.

Program Files

bigram, code, find, frcode, locate, updatedb and xargs

Descriptions

bigram

bigram is used together with code to produce older−style locate databases. To learn more about these last three programs, read the locatedb.5 manual page.

code

code is the ancestor of frcode. It was used in older−style locate databases.

find

The find program searches for files in a directory hierarchy which match a certain criteria. If no criteria is given, it lists all files in the current directory and its subdirectories.

frcode

frcode is called by updatedb to compress the list of file names using front−compression, which reduces the database size by a factor of 4 to 5.

locate

locate scans a database which contains all files and directories on a filesystem. This program lists the files and directories in this database matching a certain criteria. If a user is looking for a file this program will scan the database and tell him exactly where the files he requested are located. This only makes sense if the locate database is fairly up−to−date, else it will provide out−of−date information.

updatedb

The updatedb program updates the locate database. It scans the entire file system (including other file systems that are currently mounted unless it is told not to do so) and puts every directory and file it finds into the database that's used by the locate program, which retrieves this information. It's good practice to update this database once a day to have it up−to−date whenever it is needed.

xargs

The xargs command applies a command to a list of files. If there is a need to perform the same command on multiple files, a list can be created that names all those files (one per line) and xargs can perform that command on those files.

Findutils Installation Dependencies

Last checked against version 4.1.

Bash: sh Binutils: ar, as, ld, ranlib Diffutils: cmp Fileutils: chmod, cp, install, mv, rm
Grep: egrep, grep Gcc: cc1, collect2, cpp0, gcc Make: make Patch: patch Sed: sed
Sh−utils: basename, date, echo, hostname Textutils: cat, tr

Installing Gawk−3.1.1

```
Estimated build time:           0.39 SBU
Estimated required disk space:  15 MB
```

Installation of Gawk

This package requires its patch to be applied before you can install it. Make sure it's unpacked before running the installation commands.

Warning: do NOT run **`make uninstall`** on this package if you apply the patch to change the `libexec` directory definition. The *uninstall* rule in the `Makefile` file runs a command like **`rm -rf <libexecdir>/*`** Since we change the `libexec` directory to `/usr/bin` it'll run **`rm -rf /usr/bin/*`**

Install Gawk by running the following commands:

```
patch -Np1 -i ../gawk-3.1.1-2.patch &&
./configure --prefix=/usr --libexecdir=/usr/bin &&
make &&
make install
```

Command explanations

`patch -Np1 -i ../gawk-3.1.1-2.patch:` This patch alters the code that determines the location of the `libexec` directory. This patch will allow us to override it by passing *−−libexecdir* to the configure script.

Contents of Gawk

Last checked against version 3.1.1.

Program Files

awk, gawk, gawk−3.1.1, grcat, igawk, pgawk, pgawk−3.1.1, pwcat

Descriptions

awk

awk is a symbolic link to gawk.

gawk, gawk−3.1.1

gawk is the GNU implementation of awk, a pattern scanning and processing language.

grcat

grcat concatenates the group database, /etc/group.

igawk

igawk is a shell script which gives gawk the ability to include files.

pgawk, pgawk−3.1.1

pgawk is the profiling version of gawk.

pwcat

pwcat concatenates the password database, /etc/passwd.

Gawk Installation Dependencies

Last checked against version 3.1.0.

(No dependencies checked yet.)

Installing Ncurses–5.2

```
Estimated build time:           1.88 SBU
Estimated required disk space:  22 MB
```

Installation of Ncurses

This package requires its patch to be applied before you can install it. Make sure it's unpacked before running the installation commands.

Install Ncurses by running the following commands:

```
patch -Np1 -i ../ncurses-5.2-2.patch &&
./configure --prefix=/usr --with-shared &&
make &&
make install &&
chmod 755 /usr/lib/*.5.2 &&
mv /usr/lib/libncurses.so.5* /lib &&
ln -s libncurses.a /usr/lib/libcurses.a &&
ln -sf ../../lib/libncurses.so.5 /usr/lib/libncurses.so &&
ln -sf ../../lib/libncurses.so.5 /usr/lib/libcurses.so
```

Command explanations

`patch -Np1 -i ../ncurses-5.2-patch:` This patch fixes a compile problem with GCC–3.2 because Ncurses uses constructions that are no longer valid in the new C++ standard.

`--with-shared:` This enables the build of the shared ncurses library files.

`chmod 755 *.5.2:` Shared libraries should be executable. Ncurses's install routine doesn't set the permissions properly so we do it manually instead.

`ln -sf libncurses.a libcurses.a:` Some programs try to link using –lcurses instead of –lncurses. This symlink ensures that such programs will link without errors.

Contents of Ncurses

Last checked against version 5.2.

Program Files

captoinfo (link to tic), clear, infocmp, infotocap (link to tic), reset (link to tset), tack, tic, toe, tput and tset.

Descriptions

captoinfo

captoinfo converts a termcap description into a terminfo description.

clear

clear clears the screen if this is possible. It looks in the environment for the terminal type and then in the terminfo database to figure out how to clear the screen.

infocmp

infocmp can be used to compare a binary terminfo entry with other terminfo entries, rewrite a terminfo description to take advantage of the use= terminfo field, or print out a terminfo description from the binary file (term) in a variety of formats (the opposite of what tic does).

infotocap

info to cap converts a terminfo description into a termcap description.

reset

reset sets cooked and echo modes, turns off cbreak and raw modes, turns on new−line translation and resets any unset special characters to their default values before doing terminal initialization the same way as tset.

tack

tack is the terminfo action checker.

tic

tic is the terminfo entry−description compiler. The program translates a terminfo file from source format into the binary format for use with the ncurses library routines. Terminfo files contain information about the capabilities of a terminal.

toe

toe lists all available terminal types by primary name with descriptions.

tput

tput uses the terminfo database to make the values of terminal−dependent capabilities and information available to the shell, to initialize or reset the terminal, or return the long name of the requested terminal type.

tset

tset initializes terminals so they can be used, but it's not widely used anymore. It's provided for 4.4BSD compatibility.

Library Files

libcurses.[a,so] (link to libncurses.[a,so]), libform.[a,so], libform_g.a, libmenu.[a,so], libmenu_g.a, libncurses++.a, libncurses.[a,so], libncurses_g.a, libpanel.[a,so] and libpanel_g.a

libcurses, libncurses++, libncurses, libncurses_g

These libraries are the base of the system and are used to display text (often in a fancy way) on the screen. An example where ncurses is used is in the kernel's "make menuconfig" process.

libform, libform_g

libform is used to implement forms in ncurses.

libmenu, libmenu_g

libmenu is used to implement menus in ncurses.

libpanel, libpanel_g

libpanel is used to implement panels in ncurses.

Ncurses Installation Dependencies

Last checked against version 5.2.

Bash: sh Binutils: ar, as, ld, ranlib Diffutils: cmp Fileutils: chmod, cp, install, ln, mkdir, mv, rm
Gcc: c++, cc1, cc1plus, collect2, cpp0, gcc Glibc: ldconfig Grep: egrep, fgrep, grep Make: make
Gawk: gawk Sed: sed Sh−utils: basename, date, echo, expr, hostname, uname
Textutils: cat, sort, tr, wc

Installing Vim−6.1

```
Estimated build time:           0.81 SBU
Estimated required disk space:  24 MB
```

Installation of Vim

This package requires its patch to be applied before you can install it. Make sure it's unpacked before running the installation commands.

If you not wish to install Vim, build instructions for alternative editors are available at http://beyond.linuxfromscratch.org/view/cvs/postlfs/editors.html. Currently, there are instructions for Emacs, nano, and joe.

Install Vim by running the following commands:

```
patch -Np1 -i ../vim-6.1.patch &&
./configure --prefix=/usr &&
make CPPFLAGS=-DSYS_VIMRC_FILE=\\\"/etc/vimrc\\\" &&
make install &&
ln -s vim /usr/bin/vi
```

If you plan to install the X Window system on your LFS system, you might want to re−compile Vim after you have installed X. Vim comes with a nice GUI version of the editor which requires X and a few other libraries to be installed. For more information read the Vim documentation.

Command explanations

`patch -Np1 -i ../vim-6.1.patch`: This patch fixes a compile problem with GCC−3.2.

`make CPPFLAGS=-DSYS_VIMRC_FILE=\\\"/etc/vimrc\\\"`: Setting this will cause vim to look for the `/etc/vimrc` file that contains the global vim settings. Normally this file is looked for in `/usr/share/vim`, but `/etc` is a more logical place for this kind of file.

Contents of Vim

Last checked against version 6.1.

Program Files

efm_filter.pl, efm_perl.pl, ex (link to vim), less.sh, mve.awk, pltags.pl, ref, rview (link to vim), rvim (link to vim), shtags.pl, tcltags, vi (link to vim), view (link to vim), vim, vim132, vim2html.pl, vimdiff (link to vim), vimm, vimspell.sh, vimtutor and xxd

Descriptions

efm_filter.pl

efm_filter.pl is a filter which reads from stdin, copies to stdout and creates an error file that can be read by vim.

efm_perl.pl

efm_perl.pl reformats the error messages of the Perl interpreter for use with the quickfix mode of vim.

ex

ex starts vim in Ex mode.

less.sh

less.sh is a script which starts vim with less.vim.

mve.awk

mve.awk processes vim errors.

pltags.pl

pltags.pl creates a tags file for Perl code, for use by vim.

ref

ref checks the spelling of arguments.

rview

rview is a restricted version of view. No shell commands can be started and vim can't be suspended.

rvim

rvim is the restricted version of vim. No shell commands can be started and vim can't be suspended.

shtags.pl

shtags.pl generates a tag file for perl scripts.

tcltags

tcltags generates a tag file for TCL code.

vi

vi starts vim in vi–compatible mode.

view

view starts vim in read–only mode.

vim

vim starts vim in the normal, default way.

vim132

vim132 starts vim with the terminal in 132 column mode.

vim2html.pl

vim2html.pl converts vim documentation to HTML.

vimdiff

vimdiff edits two or three versions of a file with vim and show differences.

vimm

vimm enables the DEC locator input model on a remote terminal.

vimspell.sh

vimspell.sh is a script which spells a file and generates the syntax statements necessary to highlight in vim.

vimtutor

vimtutor starts the Vim tutor.

xxd

xxd makes a hexdump or does the reverse.

Vim Installation Dependencies

Last checked against version 6.0.

Bash: sh Binutils: as, ld, strip Diffutils: cmp, diff Fileutils: chmod, cp, ln, mkdir, mv, rm, touch Find: find Gcc: cc1, collect2, cpp0, gcc Grep: egrep, grep Make: make Net−tools: hostname Sed: sed Sh−utils: echo, expr, uname, whoami Textutils: cat, tr, wc

Installing Bison−1.35

```
Estimated build time:           0.27 SBU
Estimated required disk space:  6 MB
```

Installation of Bison

Install Bison by running the following commands:

```
./configure --prefix=/usr &&
make &&
make install
```

Some programs don't know about bison and try to find the yacc program (bison is a (better) alternative for yacc). So to please those few programs out there we'll create a bash script called yacc that calls bison and have it emulate yacc's output file name conventions.

Create a new file `/usr/bin/yacc` by running the following:

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

exec /usr/bin/bison -y "$@"

# End /usr/bin/yacc
EOF
chmod 755 /usr/bin/yacc
```

Contents of Bison

Last checked against version 1.35.

Program Files

bison and yacc

Descriptions

bison

bison is a parser generator, a replacement for yacc. yacc stands for Yet Another Compiler Compiler. What is bison then? It is a program that generates a program that analyzes the structure of a text file. Instead of writing the actual program a user specifies how things should be connected and with those rules a program is constructed that analyzes the text file. There are a lot of examples where structure is needed and one of them is the calculator.

Given the string :

1 + 2 * 3

A human can easily come to the result 7. Why? Because of the structure. Our brain knows how to interpret the string. The computer doesn't know that and bison is a tool to help it understand by presenting the string in the following way to the compiler:

```
     +         /\         *  1        /\
   2  3
```

Starting at the bottom of a tree and coming across the numbers 2 and 3 which are joined by the multiplication symbol, the computer multiplies 2 and 3. The result of that multiplication is remembered and the next thing that the computer sees is the result of 2*3 and the number 1 which are joined by the add symbol. Adding 1 to the previous result makes 7. In calculating, the most complex calculations can be broken down in this tree format and the computer just starts at the bottom and works its way up to the top and comes with the correct answer. Of course, bison isn't only used for calculators alone.

yacc

We create a bash script called yacc which calls bison using the −y option. This is for compatibility purposes for programs which use yacc instead of bison.

Bison Installation Dependencies

Last checked against version 1.31.

Bash: sh Binutils: ar, as, ld, ranlib Diffutils: cmp
Fileutils: chmod, cp, install, ln, ls, mkdir, mv, rm, rmdir Gcc: cc, cc1, collect2, cpp0, gcc
Grep: egrep, fgrep, grep Make: make Sed: sed
Sh−utils: basename, dirname, echo, expr, hostname, sleep, uname Texinfo: install−info
Textutils: cat, head, tr, uniq

Installing Less−374

```
Estimated build time:           0.13 SBU
Estimated required disk space:  2 MB
```

Installation of Less

Install Less by running the following commands:

```
./configure --prefix=/usr --bindir=/bin --sysconfdir=/etc &&
make &&
make install
```

Contents of Less

Last checked against version 374.

Program Files

less, lessecho and lesskey

Description

less

The less program is a file pager (or text viewer). It displays the contents of a file and has the ability to scroll. Less is an improvement on the common pager called "more". Less has the ability to scroll backwards through files as well and it doesn't need to read the entire file when it starts, which makes it faster when reading large files.

lessecho

lessecho is needed to expand metacharacters, such as * and ?, in filenames on Unix systems.

lesskey

lesskey is used to specify key bindings for less.

Less Installation Dependencies

Last checked against version 358.

Bash: sh Binutils: as, ld Diffutils: cmp Fileutils: chmod, install, mv, rm, touch Grep: egrep, grep Gcc: cc1, collect2, cpp0, gcc Make: make Sed: sed Sh−utils: expr, hostname, uname Textutils: cat, tr

Installing Groff−1.18

```
Estimated build time:           1.08 SBU
Estimated required disk space:  18 MB
```

Installation of Groff

Install Groff by running the following commands:

```
./configure --prefix=/usr &&
make PROCESSEDEXAMPLEFILES="" &&
make PROCESSEDEXAMPLEFILES="" install &&
ln -s soelim /usr/bin/zsoelim &&
ln -s eqn /usr/bin/geqn &&
ln -s tbl /usr/bin/gtbl
```

Command explanations

`make PROCESSEDEXAMPLEFILES=""`: Groff has a few extra dependencies that we don't install with LFS. This option disables the need for those tools.

`ln -s ...`: These symlinks are needed for some **`xman`** and other groff/man document programs to work properly.

Contents of Groff

Last checked against version 1.17.2.

Program Files

addftinfo, afmtodit, eqn, geqn (link to eqn), grn, grodvi, groff, grog, grolbp, grolj4, grops, grotty, gtbl (link to tbl), hpftodit, indxbib, lkbib, lookbib, mmroff, neqn, nroff, pfbtops, pic, post−grohtml, pre−grohtml, refer, soelim, tbl, tfmtodit, troff and zsoelim (link to soelim)

Descriptions

addftinfo

addftinfo reads a troff font file and adds some additional font−metric information that is used by the groff system.

afmtodit

afmtodit creates a font file for use with groff and grops.

eqn

eqn compiles descriptions of equations embedded within troff input files into commands that are understood by troff.

geqn

geqn is the GNU implementation of eqn.

grn

grn is a groff preprocessor for gremlin files.

grodvi

grodvi is a driver for groff that produces TeX dvi format.

groff

groff is a front–end to the groff document formatting system. Normally it runs the troff program and a post–processor appropriate for the selected device.

grog

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 prints the groff command including those options on the standard output.

grolbp

grolbp is a groff driver for Canon CAPSL printers (LBP–4 and LBP–8 series laser printers).

grolj4

grolj4 is a driver for groff that produces output in PCL5 format suitable for an HP Laserjet 4 printer.

grops

grops translates the output of GNU troff to Postscript.

grotty

grotty translates the output of GNU troff into a form suitable for typewriter–like devices.

gtbl

gtbl is the GNU implementation of tbl.

hpftodit

hpftodit creates a font file for use with groff –Tlj4 from an HP tagged font metric file.

indxbib

indxbib makes an inverted index for the bibliographic databases a specified file for use with refer, lookbib, and lkbib.

lkbib

lkbib searches bibliographic databases for references that contain specified keys and prints any references found on the standard output.

lookbib

lookbib prints a prompt on the standard error (unless the standard input is not a terminal), reads from the standard input a line containing a set of keywords, 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

mmroff is a simple preprocessor for groff.

neqn

The neqn script formats equations for ascii output.

nroff

The nroff script emulates the nroff command using groff.

pfbtops

pfbtops translates a Postscript font in .pfb format to ASCII.

pic

pic compiles descriptions of pictures embedded within troff or TeX input files into commands that are understood by TeX or troff.

pre−grohtml and post−grohtml

pre− and post−grohtml translate the output of GNU troff to html.

refer

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 about how citations are to be processed.

soelim

soelim reads files and replaces lines of the form *.so file* by the contents of *file*.

tbl

tbl compiles descriptions of tables embedded within troff input files into commands that are understood by troff.

tfmtodit

tfmtodit creates a font file for use with **`groff −Tdvi`**.

troff

troff is highly compatible with Unix troff. Usually it should be invoked using the groff command, which will also run preprocessors and post−processors in the appropriate order and with the appropriate options.

zsoelim

zsoelim is the GNU implementation of soelim.

Groff Installation Dependencies

Last checked against version 1.17.2.

Bash: sh Binutils: ar, as, ld, ranlib Bison: bison Diffutils: cmp
Fileutils: chmod, cp, install, ln, ls, mkdir, mv, rm, touch
Gcc: cc1, cc1plus, collect2, cpp0, g++, gcc Grep: egrep, grep Make: make Gawk: awk Sed: sed
Sh−utils: basename, date, echo, expr, hostname, uname Textutils: cat, tr

Installing Textutils−2.1

```
Estimated build time:           0.83 SBU
Estimated required disk space:  17 MB
```

Installation of Textutils

Install Textutils by running the following commands:

```
./configure --prefix=/usr &&
make &&
make install &&
mv /usr/bin/{cat,head} /bin
```

Contents of Textutils

Last checked against version 2.0.

Program Files

cat, cksum, comm, csplit, cut, expand, fmt, fold, head, join, md5sum, nl, od, paste, pr, ptx, sort, split, sum, tac, tail, tr, tsort, unexpand, uniq and wc

Descriptions

cat

cat concatenates file(s) or standard input to standard output.

cksum

cksum prints CRC checksum and byte counts of each specified file.

comm

comm compares two sorted files line by line.

csplit

csplit outputs pieces of a file separated by (a) pattern(s) to files xx01, xx02, ..., and outputs byte counts of each piece to standard output.

cut

cut prints selected parts of lines from specified files to standard output.

expand

expand converts tabs in files to spaces, writing to standard output.

fmt

fmt reformats each paragraph in the specified file(s), writing to standard output.

fold

fold wraps input lines in each specified file (standard input by default), writing to standard output.

head

head prints the first xx (10 by default) lines of each specified file to standard output.

join

join joins lines of two files on a common field.

md5sum

md5sum prints or checks MD5 checksums.

nl

nl writes each specified file to standard output, with line numbers added.

od

od writes an unambiguous representation, octal bytes by default, of a specified file to standard output.

paste

paste writes lines consisting of the sequentially corresponding lines from each specified file, separated by TABs, to standard output.

pr

pr paginates or columnates files for printing.

ptx

ptx produces a permuted index of file contents.

sort

sort writes sorted concatenation of files to standard output.

split

split outputs fixed−size pieces of an input file to PREFIXaa, PREFIXab, ...

sum

sum prints checksum and block counts for each specified file.

tac

tac writes each specified file to standard output, last line first.

tail

tail print the last xx (10 by default) lines of each specified file to standard output.

tr

tr translates, squeezes, and/or deletes characters from standard input, writing to standard output.

tsort

tsort writes totally ordered lists consistent with the partial ordering in specified files.

unexpand

unexpand converts spaces in each file to tabs, writing to standard output.

uniq

uniq removes duplicate lines from a sorted file.

wc

wc prints line, word and byte counts for each specified file and a total line, if more than one file is specified.

Textutils Installation Dependencies

Last checked against version 2.0.

Autoconf: autoconf, autoheader Automake: aclocal, automake Bash: sh Binutils: ar, as, ld, ranlib
Diffutils: cmp Fileutils: chmod, install, ls, mv, rm Gettext: msgfmt, xgettext
Gcc: cc, cc1, collect2, cpp0, gcc Glibc: getconf Grep: egrep, fgrep, grep M4: m4 Make: make
Gawk: gawk Net−tools: hostname Perl: perl Sed: sed Sh−utils: basename, echo, expr, sleep, uname
Tar: tar Texinfo: install−info, makeinfo Textutils: cat, tr

Installing Sed−3.02

```
Estimated build time:           0.09 SBU
Estimated required disk space:  2 MB
```

Installation of Sed

Install Sed by running the following commands:

```
./configure --prefix=/usr --bindir=/bin &&
make &&
make install
```

Contents of Sed

Last checked against version 3.02.

Program Files

sed

Descriptions

sed

sed is a stream editor. A stream editor is used to perform basic text transformations on an input stream (a file or input from a pipeline).

Sed Installation Dependencies

Last checked against version 3.02.

Autoconf: autoconf, autoheader Automake: aclocal, automake Bash: sh Binutils: ar, as, ld, ranlib Diffutils: cmp Fileutils: chmod, install, ls, mv, rm Gcc: cc1, collect2, cpp0, gcc Glibc: getconf Grep: egrep, fgrep, grep M4: m4 Make: make Gawk: gawk Sed: sed Sh−utils: echo, expr, hostname, sleep Texinfo: install−info, makeinfo Textutils: cat, tr

Installing Flex−2.5.4a

```
Estimated build time:           0.05 SBU
Estimated required disk space:  3 MB
```

Installation of Flex

Install Flex by running the following commands:

```
./configure --prefix=/usr &&
make &&
make install
```

Some programs don't know about flex and try to find the lex program (flex is a (better) alternative for lex). So to please those few programs out there we'll create a bash script called lex that calls flex and have it emulate lex.

Create a new file `/usr/bin/lex` by running the following:

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

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

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

Contents of Flex

Last checked against version 2.5.4a.

Program Files

flex, flex++ (link to flex) and lex

Descriptions

flex

flex is a tool for generating programs which recognize patterns in text. Pattern recognition is very useful in many applications. A user sets up rules about what to look for and flex will make a program that looks for those patterns. The reason people use flex is that it is much easier to set up rules for what to look for than to write the actual program which finds the text.

flex++

flex++ invokes a version of flex which is used exclusively for C++ scanners.

lex

We create a bash script called lex which calls flex using the −l option. This is for compatibility purposes for programs which use lex instead of flex.

Library Files

libfl.a

Descriptions

libfl

libfl is the flex library.

Flex Installation Dependencies

Last checked against version 2.5.4a.

Bash: sh Binutils: ar, as, ld, ranlib Bison: bison Diffutils: cmp
Fileutils: chmod, cp, install, ln, mv, rm, touch Gcc: cc1, collect2, cpp0, gcc Grep: egrep, grep
Make: make Sed: sed Sh−utils: echo, hostname Textutils: cat, tr

Installing Binutils−2.13

```
Estimated build time:           2.48 SBU
Estimated required disk space:  94 MB
```

Installation of Binutils

This package is known to behave badly when you have changed its default optimization flags (including the −march and −mcpu options). Binutils is best left alone. Therefore, if you have defined any environment variables that override default optimizations, such as CFLAGS and CXXFLAGS, we recommend unsetting or modifying them when building binutils. You have been warned.

Install Binutils by running the following commands:

```
mkdir ../binutils-build &&
cd ../binutils-build &&
../binutils-2.13/configure --prefix=/usr --enable-shared &&
make tooldir=/usr &&
make tooldir=/usr install &&
make tooldir=/usr install-info &&
cp ../binutils-2.13/include/libiberty.h /usr/include
```

Command explanations

`tooldir=/usr:` Normally, the tooldir (the directory where the executables from binutils end up) is set to $(exec_prefix)/$(target_alias) which expands into, for example, /usr/i686−pc−linux−gnu. Since we only build for our own system, we don't need this target specific directory in /usr. That setup would be used if the system was used to cross−compile (for example compiling a package on the Intel machine that generates code that can be executed on Apple PowerPC machines).

`make tooldir=/usr install-info:` This will install binutils' info pages.

`cp ../binutils-2.13/include/libiberty.h /usr/include:` The libiberty.h header file is needed in order for certain software to compile.

Contents of Binutils

Last checked against version 2.12.1.

Program Files

addr2line, ar, as, gasp, gprof, ld, nm, objcopy, objdump, ranlib, readelf, size, strings and strip

Descriptions

addr2line

addr2line translates program addresses into file names and line numbers. Given an address and an executable, it uses the debugging information in the executable to figure out which file name and line number are associated with a given address.

ar

The ar program creates, modifies, and extracts from archives. An archive is a single file holding a collection of other files in a structure that makes it possible to retrieve the original individual files (called members of the archive).

as

as is primarily intended to assemble the output of the GNU C compiler, gcc, for use by the linker ld.

gasp

gasp is the Assembler Macro Preprocessor.

gprof

gprof displays call graph profile data.

ld

ld combines a number of object and archive files, relocates their data and ties up symbol references. Often the last step in building a new compiled program to run is a call to ld.

nm

nm lists the symbols from object files.

objcopy

objcopy utility copies the contents of an object file to another. objcopy uses the GNU BFD Library to read and write the object files. It can write the destination object file in a format different from that of the source object file.

objdump

objdump displays information about one or more object files. The options control what particular information to display. This information is mostly useful to programmers who are working on the compilation tools, as opposed to programmers who just want their program to compile and work.

ranlib

ranlib generates an index to the contents of an archive, and stores it in the archive. The index lists each symbol defined by an archive member that is a relocatable object file.

readelf

readelf displays information about elf type binaries.

size

size lists the section sizes ––and the total size–– for each of the object files in its argument list. By default, one line of output is generated for each object file or each module in an archive.

strings

For each file given, strings prints the printable character sequences that are at least 4 characters long (or the number specified with an option to the program) and are followed by an unprintable character. By default, it only prints the strings from the initialized and loaded sections of object files. For other types of files, it prints the strings from the whole file.

strings is mainly useful for determining the contents of non–text files.

strip

strip discards all or specific symbols from object files. The list of object files may include archives. At least one object file must be given. strip modifies the files named in its argument, rather than writing modified copies under different names.

Library Files

libbfd.[a,so] and libopcodes.[a,so]

Descriptions

libbfd

libbfd is the Binary File Descriptor library.

libopcodes

libopcodes is a native library for dealing with opcodes and is used in the course of building utilities such as objdump. Opcodes are actually "readable text" versions of instructions for the processor.

Binutils Installation Dependencies

Last checked against version 2.11.2.

Autoconf: autoconf, autoheader Automake: aclocal, automake Bash: sh
Binutils: ar, as, ld, nm, ranlib, strip Diffutils: cmp
Fileutils: chmod, cp, ln, ls, mkdir, mv, rm, rmdir, touch Flex: flex
Gcc: cc, cc1, collect2, cpp0, gcc Glibc: ldconfig Grep: egrep, fgrep, grep M4: m4 Make: make
Gawk: gawk Sed: sed Sh–utils: basename, echo, expr, hostname, sleep, true, uname
Texinfo: install–info, makeinfo Textutils: cat, sort, tr, uniq

Installing Fileutils−4.1

```
Estimated build time:           0.68 SBU
Estimated required disk space:  17 MB
```

Installation of Fileutils

Install Fileutils by running the following commands:

```
./configure --prefix=/usr --bindir=/bin &&
make &&
make install &&
ln -s ../../bin/install /usr/bin
```

Contents of Fileutils

Last checked against version 4.1.

Program Files

chgrp, chmod, chown, cp, dd, df, dir, dircolors, du, install, ln, ls, mkdir, mkfifo, mknod, mv, rm, rmdir, shred, sync, touch and vdir

Descriptions

chgrp

chgrp changes the group ownership of each given file to the named group, which can be either a group name or a numeric group ID.

chmod

chmod changes the permissions of each given file according to mode, which can be either a symbolic representation of changes to make or an octal number representing the bit pattern for the new permissions.

chown

chown changes the user and/or group ownership of each given file.

cp

cp copies files from one place to another.

dd

dd copies a file (from the standard input to the standard output, by default) with a user−selectable blocksize, while optionally performing conversions on it.

df

df displays the amount of disk space available on the filesystem containing each file name argument. If no file name is given, the space available on all currently mounted filesystems is shown.

dir, ls and vdir

dir and vdir are versions of ls with different default output formats. These programs list each given file or directory name. Directory contents are sorted alphabetically. For ls, files are, by default, listed in columns sorted vertically if the standard output is a terminal; otherwise they are listed one per line. For dir, files are, by default, listed in columns sorted vertically. For vdir, files are, by default, listed in long format.

dircolors

dircolors outputs commands to set the LS_COLOR environment variable. The LS_COLOR variable is use to change the default color scheme used by ls and related utilities.

du

du displays the amount of disk space used by each file or directory listed on the command–line and by each of their subdirectories.

install

install copies files and sets their permission modes and, if possible, their owner and group.

ln

ln makes hard or soft (symbolic) links between files.

mkdir

mkdir creates directories with a given name.

mkfifo

mkfifo creates a FIFO with each given name.

mknod

mknod creates a FIFO, character special file or block special file with the given file name.

mv

mv moves files from one directory to another or renames files, depending on the arguments given to mv.

rm

rm removes files or directories.

rmdir

rmdir removes directories, if they are empty.

shred

shred deletes a file securely, overwriting it first so that its contents can't be recovered.

sync

sync forces changed blocks to disk and updates the super block.

touch

touch changes the access and modification times of each given file to the current time. Files that do not exist are created empty.

Fileutils Installation Dependencies

Last checked against version 4.1.

Bash: sh Binutils: ar, as, ld, ranlib Diffutils: cmp
Fileutils: chmod, cp, install, ln, ls, mkdir, mv, rm, rmdir Gettext: msgfmt, xgettext
Gcc: cc, cc1, collect2, cpp0, gcc Grep: egrep, fgrep, grep Make: make Perl: perl Sed: sed
Sh–utils: basename, echo, expr, hostname, sleep, uname Texinfo: install–info Textutils: cat, tr

Installing Sh−utils−2.0

```
Estimated build time:           0.42 SBU
Estimated required disk space:  12 MB
```

Installation of Sh−utils

This package requires its hostname−patch to be applied before you can install it. Make sure it's unpacked before running the installation commands.

Install Sh−utils by running the following commands:

```
patch −Np1 −i ../sh−utils−2.0−hostname.patch &&
./configure −−prefix=/usr &&
make &&
make install &&
mv /usr/bin/{basename,date,echo,false,pwd} /bin &&
mv /usr/bin/{sleep,stty,su,test,true,uname} /bin &&
mv /usr/bin/chroot /usr/sbin
```

FHS compliance notes

There is a command installed in this package which is named test. It is often used in shell scripts to evaluate conditions, but is more often encountered in the form of **[condition]**. These brackets are built into the bash interpreter, but the FHS dictates that there should be a `[` binary. Create it by running:

```
ln −s test /bin/[
```

Command explanations

`patch −Np1 −i ../sh−utils−2.0−hostname.patch:` This patch suppresses the build of the hostname program which we will be installed later with the net−tools package. The hostname program from the net−tools package is a much better version (and in some cases even required since it supports options that are needed by some programs such as XFree86).

Contents of Sh−utils

Last checked against version 2.0.

Program Files

basename, chroot, date, dirname, echo, env, expr, factor, false, groups, hostid, id, logname, nice, nohup, pathchk, pinky, printenv, printf, pwd, seq, sleep, stty, su, tee, test, true, tty, uname, uptime, users, who, whoami and yes

Descriptions

basename

basename strips directory and suffixes from filenames.

chroot

chroot runs a command or interactive shell with special root directory.

date

date displays the current time in a specified format, or sets the system date.

dirname

dirname strips non−directory suffixes from file name.

echo

echo displays a line of text.

env

env runs a program in a modified environment.

expr

expr evaluates expressions.

factor

factor prints the prime factors of all specified integer numbers.

false

false always exits with a status code indicating failure.

groups

groups prints a user's group memberships.

hostid

hostid prints the numeric identifier (in hexadecimal) for the current host.

id

id prints the effective user and group IDs of the current user or a given user.

logname

logname prints the current user's login name.

nice

nice runs a program with modified scheduling priority.

nohup

nohup runs a command immune to hangups, with output to a log file.

pathchk

pathchk checks whether file names are valid or portable.

pinky

pinky is a lightweight finger utility which retrieves information about a certain user.

printenv

printenv prints all or part of the environment.

printf

printf formats and prints data (the same as the C printf function).

pwd

pwd prints the name of the current/working directory.

seq

seq prints numbers in a certain range with a certain increment.

sleep

sleep delays for a specified amount of time.

stty

stty changes and prints terminal line settings.

su

su runs a shell with substitute user and group IDs.

tee

tee reads from standard input and writes to standard output and files.

test

test checks file types and compares values.

true

true always exits with a status code indicating success.

tty

tty prints the file name of the terminal connected to standard input.

uname

uname prints system information.

uptime

uptime tells how long the system has been running.

users

users prints the user names of users currently logged in to the current host.

who

who shows who is logged on.

whoami

whoami prints the user name associated with the current effective user ID.

yes

yes outputs 'y' or a given string repeatedly, until killed.

Sh−utils Installation Dependencies

Last checked against version 2.0.

Autoconf: autoconf, autoheader Automake: aclocal, automake Bash: sh Binutils: ar, as, ld, ranlib Diffutils: cmp Fileutils: chmod, chown, install, ls, mv, rm Gettext: msgfmt, xgettext Gcc: cc, cc1, collect2, cpp0, gcc Glibc: getconf Grep: egrep, fgrep, grep M4: m4 Make: make Gawk: gawk Perl: perl Sed: sed Sh−utils: basename, echo, expr, hostname, sleep, uname Tar: tar Texinfo: install−info, makeinfo Textutils: cat, tr

Installing Gettext−0.11.5

```
Estimated build time:           0.99 SBU
Estimated required disk space:  39 MB
```

Installation of Gettext

Install Gettext by running the following commands:

```
./configure --prefix=/usr &&
make &&
make install
```

Contents of Gettext

Last checked against version 0.11.2.

Program Files

config.charset, config.rpath, gettext, gettextize, hostname, msgattrib, msgcat, msgcmp, msgcomm, msgconv, msgen, msgexec, msgfilter, msgfmt, msggrep, msginit, msgmerge, msgunfmt, msguniq, ngettext, project−id, team−address, trigger, urlget, user−email and xgettext

Descriptions

config.charset

The config.charset script outputs a system−dependent table of character encoding aliases.

config.rpath

The config.rpath script outputs a system−dependent set of variables, describing how to set the run time search path of shared libraries in an executable.

gettext

The gettext package is used for internationalization (also known as i18n) and for localization (also known as l10n). Programs can be compiled with Native Language Support (NLS) which enable them to output messages in the user's native language rather than in the default English language.

gettextize

The gettextize program copies all standard gettext files into a directory. It's used to make a package with gettext translations.

hostname

The hostname program displays a network hostname in various forms.

msgattrib

The msgattrib program filters the messages of a translation catalog according to their attributes and manipulates the attributes.

msgcat

The msgcat program finds messages which are common in several raw translations.

msgcmp

The msgcmp program compares two raw translation files.

msgcomm

The msgcomm program searches messages which appear in several .po files. It's used to compare how things are translated.

msgconv

The msgconv program converts a translation catalog to a different character encoding.

msgen

The msgen program creates an English translation catalog.

msgexec

The msgexec program applies a command to all translations of a translation catalog.

msgfilter

The msgfilter program applies a filter to all translations of a translation catalog.

msgfmt

The msgfmt program compiles raw translation into machine code. It's used to create the final program/package translation file.

msggrep

The msggrep program extracts all messages of a translation catalog that match a given pattern or belong to some given source files.

msginit

The msginit program creates a new PO file, initializing the meta information with values from the user's environment.

msgmerge

The msgmerge program combines two raw translations into one file. It's used to update the raw translation with the source extract.

msgunfmt

The msgunfmt program decompiles translation files into raw translation text. It can only be used if the compiled versions are available.

msguniq

The msguniq program unifies duplicate translations in a translation catalog.

ngettext

The ngettext program displays native language translations of a textual message whose grammatical form depends on a number.

project−id

The project−id script prints a package's identification package version or package.

team−address

The team−address script prints the team's address to stdout and outputs additional instructions.

trigger

The trigger script tests whether the current package is a GNOME or KDE package.

urlget

The urlget program gets the contents of a URL.

user−email

The user−email script prints the user's email address, with confirmation from the user.

xgettext

The xgettext program extracts the message lines from the programmers' C files. It's used to make the first translation template.

Library Files

libgettextlib[a,so], libgettextsrc[a,so]

Descriptions

libgettextlib

No description is currently available.

libgettextsrc

No description is currently available.

Gettext Installation Dependencies

Last checked against version 0.10.40.

Autoconf: autoconf, autoheader Automake: aclocal, automake Bash: sh
Binutils: ar, as, ld, nm, ranlib, strip Bison: bison Diffutils: cmp
Fileutils: chmod, install, ln, ls, mkdir, mv, rm, rmdir Gcc: cc, cc1, collect2, cpp0, gcc
Grep: egrep, fgrep, grep M4: m4 Make: make Gawk: gawk Sed: sed
Sh−utils: basename, echo, expr, hostname, sleep, uname Texinfo: install−info, makeinfo
Textutils: cat, sort, tr, uniq

Installing Net−tools−1.60

```
Estimated build time:           0.16 SBU
Estimated required disk space:  5 MB
```

Installation of Net−tools

Install Net−tools by running the following commands:

```
make &&
make update
```

If you want to accept all the default answers, you can run these commands instead:

```
yes "" | make &&
make update
```

If you don't know what to answer to all the questions asked during the **make** phase, then just accept the defaults, which will be just fine in the majority of the cases. What you are asked here are a bunch of questions relating to the kind of network protocols that you have enabled in your kernel.

The default answers will enable the tools from this package to work with the most common protocols such as TCP, PPP and a bunch of others. You still need to actually enable these protocols in the kernel. What you do here is merely telling the programs to be able to use those protocols, but it's up to the kernel to make them available to the system.

Command explanations

make update: This does the same as a **make install** with the exception that make update doesn't make backups of files it's replacing. One of the things net−tools replaces is sh−utils's version of `/bin/hostname` (net−tools's version is far better than sh−utils's version).

Also, if you decide to reinstall this package at some point in the future, a **make update** won't backup all the files from a previous net−tools installation.

Contents of Net−tools

Last checked against version 1.60.

Program Files

arp, dnsdomainname (link to hostname), domainname (link to hostname), hostname, ifconfig, nameif, netstat, nisdomainname (link to hostname), plipconfig, rarp, route, slattach and ypdomainname (link to hostname)

Descriptions

arp

arp is used to manipulate the kernel's ARP cache, usually to add or delete an entry, or to dump the ARP cache.

dnsdomainname

dnsdomainname shows the system's DNS domain name.

domainname

domainname shows or sets the system's NIS/YP domain name.

hostname

hostname prints or sets the name of the current host system.

ifconfig

The ifconfig command is the general command used to configure network interfaces.

nameif

nameif names network interfaces based on MAC addresses.

netstat

netstat is a multi−purpose tool used to print the network connections, routing tables, interface statistics, masquerade connections and multicast memberships.

nisdomainname

nisdomainname shows or sets system's NIS/YP domain name.

plipconfig

plipconfig is used to fine−tune the PLIP device parameters, hopefully making it faster.

rarp

Akin to the arp program, the rarp program manipulates the system's RARP table.

route

route is the general utility which is used to manipulate the IP routing table.

slattach

slattach attaches a network interface to a serial line, i.e.. puts a normal terminal line into one of several "network" modes.

ypdomainname

ypdomainname shows or sets the system's NIS/YP domain name.

Net−tools Installation Dependencies

Last checked against version 1.60.

Bash: bash, sh Binutils: ar, as, ld Fileutils: install, ln, ls, mv, rm Gcc: cc, cc1, collect2, cpp0 Make: make Sh−utils: echo

Installing Perl−5.8.0

```
Estimated build time:           3.81 SBU
Estimated required disk space:  52 MB
```

Installation of Perl

Install Perl by running the following commands:

```
./configure.gnu --prefix=/usr &&
make &&
make install
```

If you want more control over the way perl sets itself up to be built, you can run the interactive **`Configure`** script and modify the way perl is built. If you think you can live with the (sensible) defaults perl auto−detects, then just use the commands listed above.

Contents of Perl

Last checked against version 5.6.1.

Program Files

a2p, c2ph, dprofpp, find2perl, h2ph, h2xs, perl, perl5.6.1, perlbug, perlcc, perldoc, pl2pm, pod2html, pod2latex, pod2man, pod2text, pod2usage, podchecker, podselect, pstruct, s2p and splain

Descriptions

a2p

a2p is an awk to perl translator.

c2ph

c2ph dumps C structures as generated from "cc −g −S" stabs.

dprofpp

dprofpp displays perl profile data.

find2perl

find2perl translates find command lines to Perl code.

h2ph

h2ph converts .h C header files to .ph Perl header files.

h2xs

h2xs converts .h C header files to Perl extensions.

perl, perl5.6.1

perl is the Practical Extraction and Report Language. It combines some of the best features of C, sed, awk and sh into one powerful language.

perlbug

perlbug helps to generate bug reports about perl or the modules that come with it, and mail them.

perlcc

perlcc generates executables from Perl programs.

perldoc

perldoc looks up a piece of documentation in .pod format that is embedded in the perl installation tree or in a perl script and displays it via "pod2man | nroff −man | $PAGER".

pl2pm

pl2pm is a tool to aid in the conversion of Perl4−style .pl library files to Perl5−style library modules.

pod2html

pod2html converts files from pod format to HTML format.

pod2latex

pod2latex converts files from pod format to LaTeX format.

pod2man

pod2man converts pod data to formatted *roff input.

pod2text

pod2text converts pod data to formatted ASCII text.

pod2usage

pod2usage prints usage messages from embedded pod docs in files.

podchecker

podchecker checks the syntax of pod format documentation files.

podselect

podselect prints selected sections of pod documentation on standard output.

pstruct

pstruct dumps C structures as generated from "cc −g −S" stabs.

s2p

s2p is a sed to perl translator.

splain

splain is a program to force verbose warning diagnostics in perl.

Library Files

attrs.so, B.so, ByteLoader.so, DProf.so, Dumper.so, DynaLoader.a, Fcntl.so, Glob.so, Hostname.so, IO.so, libperl.a, Opcode.so, Peek.so, POSIX.so, re.so, SDBM_File.so, Socket.so, Syslog.so and SysV.so

Descriptions

attrs

No description is currently available.

B

No description is currently available.

ByteLoader

No description is currently available.

DProf

No description is currently available.

Dumper

No description is currently available.

DynaLoader

No description is currently available.

Fcntl

No description is currently available.

Glob

No description is currently available.

Hostname

No description is currently available.

IO

No description is currently available.

libperl

No description is currently available.

Opcode

No description is currently available.

Peek

No description is currently available.

POSIX

No description is currently available.

re

No description is currently available.

SDBM_File

No description is currently available.

Socket

No description is currently available.

Syslog

No description is currently available.

SysV

No description is currently available.

Perl Installation Dependencies

Last checked against version 5.6.1.

Bash: sh Binutils: ar, as, ld, nm Diffutils: cmp Fileutils: chmod, cp, ln, ls, mkdir, mv, rm, touch
Gcc: cc, cc1, collect2, cpp0, gcc Grep: egrep, grep Make: make Gawk: awk Sed: sed
Sh−utils: basename, date, echo, expr, hostname, pwd, uname, whoami
Textutils: cat, comm, sort, split, tr, uniq, wc

Installing Linux threads−2.2.5 man pages

```
Estimated build time:           0.01 SBU
Estimated required disk space:  1.5 MB
```

Installation of Linux threads man pages

Unpack the glibc−linuxthreads package and you'll notice that you end up with two new directories. Enter the `linuxthreads` directory, not the `linuxthreads_db` directory.

Install the Linux threads man pages by running the following commands:

```
cd man &&
make &&
make install
```

Contents of Linux threads man pages

Last checked against version 2.2.5.

Support Files

Various Linux threads API manual pages.

Descriptions

Manual pages

Contains the manual pages regarding the Glibc Linux threads API.

Linux threads man pages installation Dependencies

Not yet checked.

Installing M4−1.4

```
Estimated build time:           0.08 SBU
Estimated required disk space:  3 MB
```

Installation of M4

Install M4 by running the following commands:

```
./configure --prefix=/usr &&
make &&
make install
```

Contents of M4

Last checked against version 1.4.

Program Files

m4

Descriptions

m4

m4 is a macro processor. It copies input to output, expanding macros as it goes. Macros are either built−in or user−defined and can take any number of arguments. Besides just doing macro expansion, m4 has built−in functions for including named files, running Unix commands, doing integer arithmetic, manipulating text in various ways, 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.

M4 Installation Dependencies

Last checked against version 1.4.

Bash: sh Binutils: ar, as, ld, ranlib Diffutils: cmp Fileutils: chmod, cp, install, mv, rm Make: make
Gcc: cc1, collect2, cpp0, gcc Grep: egrep, grep Sed: sed Sh−utils: date, echo, hostname
Textutils: cat, tr

Installing Texinfo−4.2

```
Estimated build time:           0.43 SBU
Estimated required disk space:  12 MB
```

Installation of Texinfo

Install Texinfo by running the following commands:

```
./configure --prefix=/usr &&
make &&
make install &&
make TEXMF=/usr/share/texmf install-tex
```

Command explanations

`make TEXMF=/usr/share/texmf install-tex:` This installs the texinfo components that belong in a TeX installation. Although TeX isn't installed on LFS, it's installed here to complete the texinfo installation.

Contents of Texinfo

Last checked against version 4.2.

Program Files

info, infokey, install−info, makeinfo, texi2dvi and texindex

Descriptions

info

The info program reads Info documents, usually contained in the /usr/share/info directory. Info documents are like man(ual) pages, but they tend to go deeper than just explaining the options to a program.

infokey

infokey compiles a source file containing Info customizations into a binary format.

install−info

The install−info program updates the info entries. When the info program is run, a list with available topics (ie: available info documents) will be presented. The install−info program is used to maintain this list of available topics. If info files are removed manually, you must also delete the topic in the index file. This program is used for that. It also works the other way around when info documents are added.

makeinfo

The makeinfo program translates Texinfo source documents into various formats. Available formats are: info files, plain text and HTML.

texi2dvi

The texi2dvi program prints Texinfo documents.

texindex

The texindex program is used to sort Texinfo index files.

Texinfo Installation Dependencies

Last checked against version 4.0.

Bash: sh Binutils: ar, as, ld, ranlib Diffutils: cmp Fileutils: chmod, install, ln, ls, mkdir, mv, rm
Gcc: cc1, collect2, cpp0, gcc Grep: egrep, fgrep, grep Make: make Sed: sed
Sh−utils: basename, echo, expr, hostname, sleep Texinfo: makeinfo Textutils: cat, tr

Installing Autoconf−2.53

```
Estimated build time:           0.05 SBU
Estimated required disk space:  6 MB
```

Installation of Autoconf

Install Autoconf by running the following commands:

```
./configure --prefix=/usr &&
make &&
make install
```

Contents of Autoconf

Last checked against version 2.53.

Program Files

autoconf, autoheader, autom4te, autoreconf, autoscan, autoupdate and ifnames

Descriptions

autoconf

autoconf is a tool for producing shell scripts that automatically configure software source code packages to adapt to many kinds of Unix−like systems. The configuration scripts produced by autoconf are independent of autoconf when they are run, so their users do not need to have autoconf.

autoheader

The autoheader program can create a template file of C #define statements for configure to use.

autom4te

autom4te runs GNU M4 on files.

autoreconf

If there are a lot of autoconf−generated configure scripts, the autoreconf program can save some work. It runs autoconf and autoheader (where appropriate) repeatedly to remake the autoconf configure scripts and configuration header templates in the directory tree rooted at the current directory.

autoscan

The autoscan program can help to create a configure.in file for a software package. autoscan examines the source files in a directory tree. If a directory is not specified on the command line, then the current working directory is used. The source files are searched for common portability problems and a configure.scan file is created to serve as the preliminary configure.in for that package.

autoupdate

The autoupdate program updates a configure.in file that calls autoconf macros by their old names to use the current macro names.

ifnames

ifnames can help when writing a configure.in for a software package. It prints the identifiers that the package already uses in C preprocessor conditionals. If a package has already been set up to have some portability, this program can help to determine what configure needs to check. It may fill in some gaps in a configure.in file generated by autoscan.

Autoconf Installation Dependencies

Last checked against version 2.52.

Bash: sh Diffutils: cmp Fileutils: chmod, install, ln, ls, mkdir, mv, rm Grep: fgrep, grep M4: m4
Make: make Gawk: gawk Sed: sed Sh−utils: echo, expr, hostname, sleep, uname Texinfo: install−info
Textutils: cat, tr

Installing Automake−1.6.3

```
Estimated build time:           0.03 SBU
Estimated required disk space:  6 MB
```

Installation of Automake

Install Automake by running the following commands:

```
./configure --prefix=/usr &&
make install
```

Contents of Automake

Last checked against version 1.6.2.

Program Files

acinstall, aclocal, aclocal−1.6, automake, automake−1.6, compile, config.guess, config.sub, depcomp, elisp−comp, install−sh, mdate−sh, missing, mkinstalldirs, py−compile, ylwrap

Descriptions

acinstall

acinstall is a script which installs aclocal−style M4 files.

aclocal, aclocal−1.6

automake includes a number of autoconf macros which can be used in packages, some of which are needed by automake in certain situations. These macros must be defined in the aclocal.m4−file or they will not be seen by autoconf.

The aclocal program will automatically generate aclocal.m4 files based on the contents of configure.in. This provides a convenient way to get automake−provided macros without having to search around. Also, the aclocal mechanism is extensible for use by other packages.

automake, automake−1.6

To create all the Makefile.in's for a package, run the automake program in the top level directory, with no arguments. automake will automatically find each appropriate Makefile.am (by scanning configure.in) and generate the corresponding Makefile.in.

compile

compile is script which acts as a wrapper for compilers.

config.guess

config.guess is a script which attempts to guess a canonical system name.

config.sub

config.sub is a configuration validation subroutine script.

depcomp

depcomp is a script which compiles a program while generating dependencies as side−effects.

elisp−comp

elisp−comp is a script which byte−compiles .el files.

install−sh

install−sh is a script which installs a program, script, or a datafile.

mdate−sh

mdate−sh is a script which prints the modification time of a file or directory.

missing

missing is a script which acts as a common stub for a few missing GNU programs during an installation.

mkinstalldirs

mkinstalldirs is a script which makes a directory hierarchy.

py−compile

py−compile is a script which compiles a Python program.

ylwrap

ylwrap is a script which acts as a wrapper for lex/yacc invocations.

Automake Installation Dependencies

Last checked against version 1.5.

Bash: sh Diffutils: cmp Fileutils: chmod, install, ls, mkdir, mv, rm, rmdir Grep: fgrep, grep Make: make Perl: perl Sed: sed Sh−utils: echo, expr, hostname, sleep Texinfo: install−info Textutils: cat, tr

Installing Bash−2.05a

```
Estimated build time:           0.82 SBU
Estimated required disk space:  14 MB
```

Installation of Bash

Install Bash by running the following commands:

```
./configure --prefix=/usr --bindir=/bin &&
make &&
make install &&
exec /bin/bash --login
```

Contents of Bash

Last checked against version 2.05a.

Program Files

bash, sh (link to bash) and bashbug

Descriptions

bash

bash is the Bourne−Again SHell, which is a widely used command interpreter on Unix systems. The bash program reads from standard input, the keyboard. A user types something and the program will evaluate what he has typed and do something with it, like running a program.

bashbug

bashbug is a shell script to help the user compose and mail bug reports concerning bash in a standard format.

sh

sh is 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.

Bash Installation Dependencies

Last checked against version 2.05a.

Bash: bash, sh Binutils: ar, as, ld, ranlib, size Diffutils: cmp
Fileutils: chmod, cp, install, ln, ls, mkdir, mv, rm Gcc: cc, cc1, collect2, cpp0, gcc
Grep: egrep, grep Make: make Gawk: awk Sed: sed
Sh−utils: basename, echo, expr, hostname, sleep, uname Texinfo: install−info Textutils: cat, tr, uniq

Installing File−3.39

```
Estimated build time:           0.21 SBU
Estimated required disk space:  2 MB
```

Installation of File

Install File by running the following commands:

```
./configure --prefix=/usr --datadir=/usr/share/misc &&
make &&
make install
```

Contents of File

Last checked against version 3.39.

Program Files

file

Descriptions

file

file tests each specified file in an attempt to classify it. There are three sets of tests, performed in this order: filesystem tests, magic number tests and language tests. The first test that succeeds causes the file type to be printed.

File Installation Dependencies

Last checked against version 3.37.

Autoconf: autoconf, autoheader Automake: aclocal, automake Bash: sh Binutils: as, ld Diffutils: cmp Fileutils: chmod, install, ln, ls, mv, rm, touch Gcc: cc1, collect2, cpp0, gcc Grep: egrep, grep M4: m4 Make: make Gawk: gawk Sed: sed Sh−utils: echo, expr, hostname, sleep Texinfo: makeinfo Textutils: cat, tr

Installing Libtool−1.4.2

```
Estimated build time:           0.15 SBU
Estimated required disk space:  7 MB
```

Installation of Libtool

Install Libtool by running the following commands:

```
./configure --prefix=/usr &&
make &&
make install
```

Contents of Libtool

Last checked against version 1.4.2.

Program Files

libtool and libtoolize

Descriptions

libtool

libtool provides generalized library−building support services.

libtoolize

libtoolize provides a standard way to add libtool support to a package.

Library Files

libltdl.a, libltdl.so (link to libltdl.so.3.1.0), libltdl.so.3 (link to libltdl.so.3.1.0) and libltdl.so.3.1.0

Descriptions

libltdl, libltdl.so.3, libltdl.so.3.1.0

A small library that aims at hiding, from programmers, the various difficulties of dlopening libraries.

Libtool Installation Dependencies

Last checked against version 1.4.2.

Bash: sh Binutils: ar, as, ld, nm, ranlib, strip Diffutils: cmp
Fileutils: chmod, cp, install, ln, ls, mkdir, mv, rm, rmdir Gcc: cc, cc1, collect2, cpp0
Glibc: ldconfig Grep: egrep, fgrep, grep Make: make Sed: sed
Sh−utils: echo, expr, hostname, sleep, uname Texinfo: install−info Textutils: cat, sort, tr, uniq

Installing Bin86−0.16.3

```
Estimated build time:           0.07 SBU
Estimated required disk space:  2 MB
```

Installation of Bin86

This package is only needed if you decide to use Lilo on your LFS system. If you're going to use something else like Grub you won't need bin86. Check the documentation for your favorite boot loader to see if you need the bin86 package (usually only ld86 and/or as86 from this package are required).

Keep in mind, though, that it's not just boot loaders that use the bin86 package. There is always the chance that some other package needs programs from this package, so keep that in mind if you decide to skip this.

Install Bin86 by running the following commands:

```
make &&
make PREFIX=/usr install
```

Contents of Bin86

Last checked against version 0.16.3

Program Files

as86, as86_encap, ld86, nm86 (link to objdump86), objdump86 and size86 (link to objdump86)

Descriptions

as86

as86 is an assembler for the 8086...80386 processors.

as86_encap

as86_encap is a shell script to call as86 and convert the created binary into a C file prog.v to be included in or linked with programs like boot block installers.

ld86

ld86 understands only the object files produced by the as86 assembler. It can link them into either an impure or a separate I&D executable.

nm86

The symbol table of the binary file.

objdump86

Dumps detailed information about a binary file.

size86

Summary sizes of the data in a binary file.

Bin86 Installation Dependencies

Last checked against version 0.16.0.

Bash: sh Binutils: as, ld, strip Fileutils: chmod, install, ln, mv Gcc: cc, cc1, collect2, cpp0 Make: make Sed: sed

Installing Bzip2−1.0.2

```
Estimated build time:           0.09 SBU
Estimated required disk space:  3 MB
```

Installation of Bzip2

Install Bzip2 by running the following commands:

```
make −f Makefile−libbz2_so &&
make &&
make install &&
cp bzip2−shared /bin/bzip2 &&
ln −s libbz2.so.1.0 libbz2.so &&
cp −a libbz2.so* /lib &&
rm /lib/libbz2.so &&
ln −s ../../lib/libbz2.so.1.0 /usr/lib/libbz2.so &&
rm /usr/bin/{bunzip2,bzcat,bzip2} &&
mv /usr/bin/{bzip2recover,bzless,bzmore} /bin &&
ln −s bzip2 /bin/bunzip2 &&
ln −s bzip2 /bin/bzcat
```

Although it's not strictly a part of a basic LFS system, it's worth mentioning that a patch for Tar can be downloaded which enables the tar program to compress and uncompress using bzip2/bunzip2 easily. With a plain tar, you have to use constructions like **`bzcat file.tar.bz | tar −xv`** or **`tar −−use−compress−prog=bunzip2 −xvf file.tar.bz2`** to use bzip2 and bunzip2 with tar. This patch provides the **`−j`** option so you can unpack a bzip2'ed archive with **`tar −xvfj file.tar.bz2`**. Applying this patch will be mentioned later on when the Tar package is re−installed.

Command explanations

`make −f Makefile−libbz2_so:` This will cause bzip2 to be built using a different Makefile file, in this case the Makefile−libbz2_so file, which creates a dynamic libbz2.so library and links the bzip2 utilities against it.

Contents of Bzip2

Last checked against version 1.0.2

Program Files

bunzip2 (link to bzip2), bzcat (link to bzip2), bzcmp, bzdiff, bzegrep, bzfgrep, bzgrep, bzip2, bzip2recover, bzless and bzmore

Descriptions

bunzip2

bunzip2 decompresses files that are compressed with bzip2.

bzcat

bzcat (or bzip2 −dc) decompresses all specified files to the standard output.

bzcmp, bzdiff

bzcmp and bzdiff are used to invoke the cmp or the diff program on bzip2 compressed files.

bzegrep, bzfgrep, bzgrep

bzegrep, bzfgrep, and bzgrep invoke either egrep, fgrep, or grep (respectively) on bzip2−compressed files.

bzip2

bzip2 compresses files using the Burrows−Wheeler block sorting text compression algorithm and Huffman coding. Compression is generally considerably better than that achieved by more conventional LZ77/LZ78−based compressors and approaches the performance of the PPM family of statistical compressors.

bzip2recover

bzip2recover recovers data from damaged bzip2 files.

bzless

bzless is a filter which allows examination of compressed or plain text files, one screenful at a time on a soft−copy terminal, like less.

bzmore

bzmore is a filter which allows examination of compressed or plain text files, one screenful at a time on a soft−copy terminal, like more.

Library Files

libbz2.a, libbz2.so (link to libbz2.so.1.0), libbz2.so.1.0 (link to libbz2.so.1.0.2) and libbz2.so.1.0.2

libbz2

libbz2 is the library for implementing lossless, block−sorting data compression, using the Burrows−Wheeler algorithm.

Bzip2 Installation Dependencies

Last checked against version 1.0.1.

Bash: sh Binutils: ar, as, ld, ranlib Fileutils: cp, ln, rm Gcc: cc1, collect2, cpp0, gcc Make: make

Installing Ed−0.2

```
Estimated build time:           0.06 SBU
Estimated required disk space:  3 MB
```

Installation of Ed

This package requires its patch to be applied before you can install it. Make sure it's unpacked before running the installation commands.

Ed isn't something you would personally use. It's installed here because it can be used by the patch program if you encounter an ed−based patch file. This happens rarely because diff−based patches are preferred these days.

Install Ed by running the following commands:

```
patch −Np1 −i ../ed−0.2.patch &&
./configure −−prefix=/usr &&
make &&
make install &&
mv /usr/bin/{ed,red} /bin
```

Command explanations

`patch −Np1 −i ../ed−0.2.patch:` This patch fixes a symlink vulnerability in ed. The ed executable creates files in /tmp with predictable names. By using various symlink attacks, it is possible to have ed write to files it should not, change the permissions of files, etc.

Contents of Ed

Last checked against version 0.2.

Program Files

ed and red (link to ed)

Description

ed

ed is a line−oriented text editor. It is used to create, display, modify and otherwise manipulate text files.

red

red is a restricted ed: it can only edit files in the current directory and cannot execute shell commands.

Ed Installation Dependencies

Last checked against version 0.2.

Bash: sh Binutils: ar, as, ld, ranlib Diffutils: cmp Fileutils: chmod, cp, install, ln, mv, rm, touch Gcc: cc1, collect2, cpp0, gcc Grep: egrep, grep Make: make Sed: sed Sh−utils: hostname Textutils: cat, tr

Installing Kbd−1.06

```
Estimated build time:           0.12 SBU
Estimated required disk space:  8 MB
```

Installation of Kbd

This package requires its patch to be applied before you can install it. Make sure it's unpacked before running the installation commands.

Install Kbd by running the following commands:

```
patch -Np1 -i ../kbd-1.06-3.patch &&
./configure &&
make &&
make install
```

Command explanations

`patch -Np1 -i ../kbd-1.06-3.patch`: This patch fixes two problems. The first one is the **`loadkeys -d`** behaviour, which is broken in current kbd versions. It is necessary to fix this, because the boot scripts rely on a proper **`loadkeys -d`**. The second part of the patch changes a Makefile so some utilities that are not installed by default (setlogcons, setvesablank and getunimap) are also installed.

Contents of Kbd

Last checked against version 1.06.

Program Files

chvt, deallocvt, dumpkeys, fgconsole, getkeycodes, getunimap, kbd_mode, kbdrate, loadkeys, loadunimap, mapscrn, openvt, psfaddtable (link to psfxtable), psfgettable (link to psfxtable), psfstriptable (link to psfxtable), psfxtable, resizecons, setfont, setkeycodes, setleds, setlogcons, setmetamode, setvesablank, showfont, showkey, unicode_start, and unicode_stop

Descriptions

chvt

chvt changes foreground virtual terminal.

deallocvt

deallocvt deallocates unused virtual terminals.

dumpkeys

dumpkeys dumps keyboard translation tables.

fgconsole

fgconsole prints the number of the active virtual terminal.

getkeycodes

getkeycodes prints the kernel scancode−to−keycode mapping table.

getunimap

getunimap prints the currently used unimap.

kbd_mode

kbd_mode reports or sets the keyboard mode.

kbdrate

kbdrate sets the keyboard repeat and delay rates.

loadkeys

loadkeys loads keyboard translation tables.

loadunimap

loadunimap loads the kernel unicode−to−font mapping table.

mapscrn

mapscrn loads a user defined output character mapping table into the console driver. Note that it is obsolete and that its features are built into setfont.

openvt

openvt starts a program on a new virtual terminal (VT).

psfaddtable, psfgettable, psfstriptable, psfxtable

These are a set of tools for handling Unicode character tables for console fonts.

resizecons

resizecons changes the kernel idea of the console size.

setfont

This lets you change the EGA/VGA fonts in console.

setkeycodes

setkeycodes loads kernel scancode−to−keycode mapping table entries.

setleds

setleds sets the keyboard LEDs. Many people find it useful to have numlock enabled by default and, by using this program, you can achieve this.

setlogcons

setlogcons sends kernel messages to the console.

setmetamode

setmetamode defines the keyboard meta key handling.

setvesablank

This lets you fiddle with the built−in hardware screensaver (not toasters, only a blank screen).

showfont

showfont displays data about a font. The information shown includes font information, font properties, character metrics and character bitmaps.

showkey

showkey examines the scancodes and keycodes sent by the keyboard.

unicode_start

unicode_start puts the console in Unicode mode.

unicode_stop

unicode_stop reverts keyboard and console from unicode mode.

Kbd Installation Dependencies

Last checked against version 1.06.

Bash: sh Binutils: as, ld, strip Bison: bison Diffutils: cmp Fileutils: cp, install, ln, mv, rm Flex: flex Gettext: msgfmt, xgettext Gcc: cc1, collect2, cpp0, gcc Grep: grep Gzip: gunzip, gzip Make: make Patch: patch Sed: sed Sh−utils: uname

Installing Diffutils−2.8.1

```
Estimated build time:           0.31 SBU
Estimated required disk space:  6 MB
```

Installation of Diffutils

Install Diffutils by running the following commands:

```
./configure --prefix=/usr &&
make &&
make install
```

Contents of Diffutils

Last checked against version 2.8.1.

Program Files

cmp, diff, diff3 and sdiff

Descriptions

cmp and diff

cmp and diff both compare two files and report their differences. Both programs have extra options which compare files in different situations.

diff3

The difference between diff and diff3 is that diff compares 2 files, diff3 compares 3 files.

sdiff

sdiff merges two files and interactively outputs the results.

Diffutils Installation Dependencies

Last checked against version 2.7.

Bash: sh Binutils: ld, as Diffutils: cmp Fileutils: chmod, cp, install, mv, rm
Gcc: cc1, collect2, cpp0, gcc Grep: egrep, grep Make: make Sed: sed Sh−utils: date, hostname
Textutils: cat, tr

Installing E2fsprogs−1.27

```
Estimated build time:           0.80 SBU
Estimated required disk space:  13 MB
```

Installation of E2fsprogs

Install E2fsprogs by running the following commands:

```
mkdir ../e2fsprogs-build &&
cd ../e2fsprogs-build &&
../e2fsprogs-1.27/configure --prefix=/usr --with-root-prefix="" \
    --enable-elf-shlibs &&
make &&
make install &&
make install-libs &&
install-info /usr/share/info/libext2fs.info /usr/share/info/dir
```

Command explanations

`--with-root-prefix="":` The reason for supplying this option is because of the setup of the e2fsprogs Makefile. Some programs are essential for system use when, for example, /usr isn't mounted (like the e2fsck program). These programs and libraries, therefore, belong in directories like /lib and /sbin. If this option isn't passed to E2fsprogs's configure, it places these programs in /usr, which is not what we want.

`--enable-elf-shlibs:` This creates shared libraries that some programs in this package can make use of.

`make install-libs:` This installs the shared libraries that are built.

Contents of E2fsprogs

Last checked against version 1.27.

Program Files

badblocks, chattr, compile_et, debugfs, dumpe2fs, e2fsck, e2image, e2label, fsck, fsck.ext2, fsck.ext3, lsattr, mk_cmds, mke2fs, mkfs.ext2, mkfs.ext3, mklost+found, resize2fs, tune2fs and uuidgen

Descriptions

badblocks

badblocks is used to search for bad blocks on a device (usually a disk partition).

chattr

chattr changes the file attributes on a Linux second extended file system.

compile_et

compile_et is used to convert a table, listing error−code names and associated messages, into a C source file that is suitable for use with the com_err library.

debugfs

The debugfs program is a file system debugger. It can be used to examine and change the state of an ext2 file system.

dumpe2fs

dumpe2fs prints the super block and blocks group information for the filesystem present on a specified device.

e2fsck and fsck.ext2

e2fsck and fsck.ext2 are used to check, and optionally repair, Linux second extended filesystems.

e2image

e2image is used to save critical ext2 filesystem data to a file.

e2label

e2label will display or change the filesystem label on the ext2 filesystem located on the specified device.

fsck

fsck is used to check, and optionally repair, a Linux file system.

fsck.ext3

fsck.ext3 is used to check, and optionally repair, a Linux ext3 filesystems.

lsattr

lsattr lists the file attributes on a second extended file system.

mk_cmds

The mk_cmds utility takes a command table file as input and produces a C source file as output, which is intended to be used with the subsystem library, libss.

mke2fs and mkfs.ext2

mke2fs is used to create a Linux second extended file system on a device (usually a disk partition). mkfs.ext2 does the same as mke2fs.

mkfs.ext3

mkfs.ext3 is used to create an ext3 filesystem.

mklost+found

mklost+found is used to create a lost+found directory in the current working directory on a Linux second extended file system. mklost+found pre−allocates disk blocks to the directory to make it usable by e2fsck.

resize2fs

resize2fs is used to resize ext2 file systems.

tune2fs

tune2fs adjusts tunable filesystem parameters on a Linux second extended filesystem.

uuidgen

The uuidgen program creates a new universally unique identifier (UUID) using the libuuid library. The 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.

Library Files

libcom_err.[a,so], libe2p.[a,so], libext2fs.[a,so], libss.[a,so], libuuid.[a,so]

Descriptions

libcom_err

The common error display routine.

libe2p

libe2p is used by dumpe2fs, chattr, and lsattr.

libext2fs

The ext2fs library is designed to allow user−level programs to manipulate an ext2 filesystem.

libss

libss is used by debugfs.

libuuid

The libuuid library is used to generate unique identifiers for objects that may be accessible beyond the local system.

E2fsprogs Installation Dependencies

Last checked against version 1.25.

Bash: sh Binutils: ar, as, ld, ranlib, strip Diffutils: cmp
Fileutils: chmod, cp, install, ln, mkdir, mv, rm, sync Gcc: cc, cc1, collect2, cpp0 Glibc: ldconfig
Grep: egrep, grep Gzip: gzip Make: make Gawk: awk Sed: sed
Sh−utils: basename, echo, expr, hostname, uname Texinfo: makeinfo Textutils: cat, tr

Installing Grep−2.5

```
Estimated build time:           0.22 SBU
Estimated required disk space:  5 MB
```

Installation of Grep

Install Grep by running the following commands:

```
./configure --prefix=/usr --bindir=/bin &&
make &&
make install
```

Contents of Grep

Last checked against version 2.5.

Program Files

egrep (link to grep), fgrep (link to grep) and grep

Descriptions

egrep

egrep prints lines from files matching an extended regular expression pattern.

fgrep

fgrep prints lines from files matching a list of fixed strings, separated by newlines, any of which is to be matched.

grep

grep prints lines from files matching a basic regular expression pattern.

Grep Installation Dependencies

Last checked against version 2.4.2.

Autoconf: autoconf, autoheader Automake: aclocal, automake Bash: sh Binutils: as, ld Diffutils: cmp
Fileutils: chmod, install, ls, mkdir, mv, rm Gettext: msgfmt, xgettext
Gcc: cc, cc1, collect2, cpp0, gcc Glibc: getconf Grep: egrep, fgrep, grep M4: m4 Make: make
Gawk: gawk Sed: sed Sh−utils: basename, echo, expr, hostname, sleep, uname
Texinfo: install−info, makeinfo Textutils: cat, tr

Installing Gzip–1.2.4a

```
Estimated build time:           0.03 SBU
Estimated required disk space:  2 MB
```

Installation of Gzip

Install Gzip by running the following commands:

```
patch −Np1 −i ../gzip−1.2.4b.patch &&
./configure −−prefix=/usr &&
cp gzexe.in{,.backup} &&
sed 's%"BINDIR"%/bin%' gzexe.in.backup > gzexe.in &&
make &&
make install &&
mv /usr/bin/gzip /bin &&
rm /usr/bin/{gunzip,zcat} &&
ln −s gzip /bin/gunzip &&
ln −s gzip /bin/zcat &&
ln −s gunzip /bin/uncompress
```

Command explanations

`patch −Np1 −i ../gzip−1.2.4b.patch:` This patch fixes a buffer overflow that occurs when a filename is longer than 1020 characters.

Contents of Gzip

Last checked against version 1.2.4a.

Program Files

gunzip (link to gzip), gzexe, gzip, uncompress (link to gunzip), zcat (link to gzip), zcmp, zdiff, zforce, zgrep, zmore and znew

Description

gunzip, uncompress

gunzip and uncompress decompress files which are compressed with gzip.

gzexe

gzexe allows you to compress executables in place and have them automatically uncompress and execute when they are run (at a penalty in performance).

gzip

gzip reduces the size of the named files using Lempel–Ziv coding (LZ77).

zcat

zcat uncompresses, and writes to standard output, either a list of files on the command line or a file being read from standard input.

zcmp

zcmp invokes the cmp program on compressed files.

zdiff

zdiff invokes the diff program on compressed files.

zforce

zforce forces a .gz extension on all gzip files so that gzip will not compress them twice. This can be useful for files with names truncated after a file transfer.

zgrep

zgrep invokes the grep program on compressed files.

zmore

zmore is a filter which allows examination of compressed or plain text files, one screen at a time on a soft−copy terminal (similar to the more program).

znew

znew re−compresses files from .Z (compress) format to .gz (gzip) format.

Gzip Installation Dependencies

Last checked against version 1.2.4a.

Bash: sh Binutils: as, ld, nm Fileutils: chmod, cp, install, ln, mv, rm
Gcc: cc1, collect2, cpp, cpp0, gcc Grep: egrep, grep Make: make Sed: sed Sh−utils: hostname
Textutils: cat, tr

Installing Man−1.5k

```
Estimated build time:           0.05 SBU
Estimated required disk space:  2 MB
```

Installation of Man

Run the following commands to install man:

```
patch −Np1 −i ../man−1.5k.patch &&
PATH=$PATH:/usr/bin:/bin \
    ./configure −default −confdir=/etc &&
make &&
make install
```

Note: If you wish to disable SGR escape sequences, you should edit the man.conf file and add the **−c** argument to nroff.

You may want to take a look at the man hint at http://hints.linuxfromscratch.org/hints/man.txt, which deals with formatting and compression issues for man pages.

Command explanations

PATH=$PATH:/usr/bin:/bin ./configure −default: The paths to some programs get written into man's files. Unfortunately, the configure script picks the last location in PATH rather than the first where a program is found. By appending /usr/bin:/bin to PATH for the ./configure command, we make sure that man doesn't use the /static versions of our programs.

patch −Np1 −i ../man−1.5k.patch: This patch comments out one of the files in the `man.conf` file (MANPATH /usr/man) because it will create redundant results when using programs like **whatis**. It also adds the *−R* option to the *PAGER* variable so man pages are displayed properly.

Contents of Man

Last checked against version 1.5k.

Program Files

apropos, makewhatis, man, man2dvi, man2html and whatis

Descriptions

apropos

apropos searches for keywords in a set of database files, containing short descriptions of system commands, and displays the result on the standard output.

makewhatis

makewhatis reads all the manual pages contained in given sections of manpath or the pre−formatted pages contained in the given sections of catpath. For each page, it writes a line in the whatis database. Each line consists of the name of the page and a short description, separated by a dash. The description is extracted using the content of the NAME section of the manual page.

man

man formats and displays the on−line manual pages.

man2dvi

man2dvi converts a manual page into dvi format.

man2html

man2html converts a manual page into html.

whatis

whatis searches for keywords in a set of database files, containing short descriptions of system commands, and displays the result on the standard output. Only complete word matches are displayed.

Man Installation Dependencies

Last checked against version 1.5i2.

Bash: sh Binutils: as, ld Fileutils: chmod, cp, install, mkdir, rm Gcc: c11, collect2, cpp0, gcc
Grep: grep Make: make Gawk: awk Sed: sed Sh−utils: echo Textutils: cat

Installing Lilo−22.2

```
Estimated build time:           0.08 SBU
Estimated required disk space:  3 MB
```

Installation of Lilo

We have chosen Lilo as a boot loader because we feel comfortable with it, but you may wish to choose another. Fabio Fracassi has written a hint on GRUB, which is available at http://hints.linuxfromscratch.org/hints/grub−howto.txt.

Install Lilo by running the following commands:

```
make &&
make install
```

It appears that compilation of this package fails on certain machines when the −g compiler flag is used. If you can't compile Lilo at all, you should try to remove the −g value from the CFLAGS variable in the `Makefile` file.

At the end of the installation the make install process will print a message stating that /sbin/lilo has to be executed to complete the update. Don't do this, as it has no use: the /etc/lilo.conf isn't present yet. We will complete the installation of lilo in Chapter 8.

The standard LILO prompt, or menu, may be replaced by the LFS logo or any logo you like. Martin Imobersteg has written a hint about this, which is located at http://hints.linuxfromscratch.org/hints/bootlogo.txt.

Contents of Lilo

Last checked against version 22.2.

Program Files

lilo, mkrescue and keytab−lilo.pl

Descriptions

lilo

lilo installs the Linux boot loader which is used to start a Linux system.

mkrescue

mkrescue makes a bootable rescue floppy using the existing kernel and any initial ramdisk.

keytab−lilo.pl

keytab−lilo.pl compiles keytable definitions into a format which lilo can use in order to set the keyboard type during boot.

Lilo Installation Dependencies

Last checked against version 22.1.

Bash: sh Bin86: as86, ld86 Binutils: as, ld, strip Fileutils: cp, dd, ln Gcc: cc, cc1, collect2, cpp0
Make: make Sed: sed Textutils: cat

Installing Make−3.79.1

```
Estimated build time:           0.22 SBU
Estimated required disk space:  6 MB
```

Installation of Make

Install Make by running the following commands:

```
./configure --prefix=/usr &&
make &&
make install &&
chgrp root /usr/bin/make &&
chmod 755 /usr/bin/make
```

Command explanations

By default `/usr/bin/make` is installed setgid kmem. This is needed on some systems so it can check the load average by using `/dev/kmem`. However, on Linux systems, setgid kmem is not needed, so we remove this from our `make` binary. This also fixes problems with the `make` ignoring certain variables like *LD_LIBRARY_PATH*.

Contents of Make

Last checked against version 3.79.1.

Program files

make

Descriptions

make

make determines automatically which pieces of a large program need to be recompiled, and issues the commands to recompile them.

Make Installation Dependencies

Last checked against version 3.79.1.

Autoconf: autoconf, autoheader Automake: aclocal, automake Bash: sh Binutils: as, ld Diffutils: cmp
Fileutils: chgrp, chmod, install, ls, mv, rm Gcc: cc, cc1, collect2, cpp0, gcc Glibc: getconf
Grep: egrep, fgrep, grep M4: m4 Make: make Gawk: gawk Sed: sed
Sh−utils: basename, echo, expr, hostname, sleep, uname Texinfo: install−info, makeinfo
Textutils: cat, tr

Installing Modutils−2.4.19

```
Estimated build time:           0.13 SBU
Estimated required disk space:  3 MB
```

Installation of Modutils

Install Modutils by running the following commands:

```
./configure &&
make &&
make install
```

Contents of Modutils

Last checked against version 2.4.16.

Program Files

depmod, genksyms, insmod, insmod_ksymoops_clean, kallsyms (link to insmod), kernelversion, ksyms (link to insmod), lsmod (link to insmod), modinfo, modprobe (link to insmod) and rmmod (link to insmod)

Descriptions

depmod

depmod handles dependency descriptions for loadable kernel modules.

genksyms

genksyms reads (on standard input) the output from gcc −E source.c and generates a file containing version information.

insmod

insmod installs a loadable module in the running kernel.

insmod_ksymoops_clean

insmod_ksymoops_clean deletes saved ksyms and modules not accessed in 2 days.

kallsyms

kallsyms extracts all kernel symbols for debugging.

kernelversion

kernelversion reports the major version of the running kernel.

ksyms

ksyms displays exported kernel symbols.

lsmod

lsmod shows information about all loaded modules.

modinfo

modinfo examines an object file associated with a kernel module and displays any information that it can glean.

modprobe

modprobe uses a Makefile–like dependency file, created by depmod, to automatically load the relevant module(s) from the set of modules available in predefined directory trees.

rmmod

rmmod unloads loadable modules from the running kernel.

Modutils Installation Dependencies

Last checked against version 2.4.12.

Bash: sh Binutils: ar, as, ld, ranlib, strip Bison: bison Diffutils: cmp
Fileutils: chmod, install, ln, mkdir, mv, rm Flex: flex Gcc: cc, cc1, collect2, cpp0, gcc
Grep: egrep, grep Make: make Sed: sed Sh–utils: basename, expr, hostname, uname Textutils: cat, tr

Installing Netkit−base−0.17

```
Estimated build time:           0.03 SBU
Estimated required disk space:  1 MB
```

Installation of Netkit−base

Install Netkit−base by running the following commands:

```
./configure &&
make &&
make install &&
cp etc.sample/{services,protocols} /etc
```

There are other files in the `etc.sample` directory which might be of interest to you.

Contents of Netkit−base

Last checked against version 0.17.

Program Files

inetd and ping

Descriptions

inetd

inetd is the mother of all daemons. It listens for connections, and transfers the call to the appropriate daemon.

ping

ping sends ICMP ECHO_REQUEST packets to a host and determines its response time.

Netkit−base Installation Dependencies

Last checked against version 0.17.

Bash: sh Binutils: as, ld, strip Fileutils: cp, install, rm Make: make Gcc: cc1, collect2, cpp0, gcc
Sed: sed Sh−utils: date Textutils: cat

Installing Patch−2.5.4

```
Estimated build time:           0.10
Estimated required disk space:  2 MB
```

Installation of Patch

Install Patch by running the following commands:

```
CPPFLAGS=-D_GNU_SOURCE \
    ./configure --prefix=/usr &&
make &&
make install
```

Contents of Patch

Last checked against version 2.5.4.

Program Files

patch

Descriptions

patch

The patch program modifies a file according to a patch file. A patch file usually is a list created by the diff program that contains instructions on how an original file needs to be modified. Patch is used a lot for source code patches since it saves time and space. Imagine a package that is 1 MB in size. The next version of that package only has changes in two files of the first version. It can be shipped as an entirely new package of 1 MB or just as a patch file of 1 KB which will update the first version to make it identical to the second version. So if the first version was downloaded already, a patch file avoids a second large download.

Patch Installation Dependencies

Last checked against version 2.5.4.

Bash: sh Binutils: as, ld Diffutils: cmp Fileutils: chmod, install, mv, rm
Gcc: cc, cc1, collect2, cpp0, gcc Glibc: getconf Grep: egrep, grep Make: make Sed: sed
Sh−utils: echo, expr, hostname, uname Textutils: cat, tr

Installing Procinfo−18

```
Estimated build time:           0.02 SBU
Estimated required disk space:  168 KB
```

Installation of Procinfo

Install Procinfo by running the following commands:

```
make LDLIBS=-lncurses &&
make install
```

Command explanations

make LDLIBS=-lncurses : This will use −lncurses instead of −ltermcap when building procinfo. This is done because libtermcap is declared obsolete in favor of libncurses.

Contents of Procinfo

Last checked against version 18.

Program Files

lsdev, procinfo and socklist

Descriptions

lsdev

lsdev gathers information about your computer's installed hardware from the interrupts, ioports and dma files in the /proc directory, thus giving you a quick overview of which hardware uses what I/O addresses and what IRQ and DMA channels.

procinfo

procinfo gathers some system data from the /proc directory and prints it nicely formatted on the standard output device.

socklist

is a Perl script that gives you a list of all open sockets, enumerating types, port, inode, uid, pid, fd and the program to which it belongs.

Procinfo Installation Dependencies

Last checked against version 18.

Binutils: as, ld Fileutils: install, mkdir Gcc: cc1, collect2, cpp0, gcc Make: make

Installing Procps−2.0.7

```
Estimated build time:           0.14 SBU
Estimated required disk space:  2 MB
```

Installation of Procps

This package requires its patch to be applied before you can install it. Make sure it's unpacked before running the installation commands.

Install Procps by running the following commands:

```
patch −Np1 −i ../procps−2.0.7.patch &&
make &&
make XSCPT="" install &&
mv /usr/bin/kill /bin
```

Command explanations

`patch −Np1 −i ../procps−2.0.7.patch:` This patch fixes a locale problem that makes top crash under certain locale settings.

`make XSCPT="" install:` This will set the Makefile variable XSCPT to an empty value so that the XConsole installation is disabled. Otherwise "Make install" tries to copy the file XConsole to /usr/X11R6/lib/X11/app−defaults. And that directory does not exist, because X is not installed.

Contents of Procps

Last checked against version 2.0.7.

Program Files

free, kill, oldps, pgrep, pkill, ps, skill, snice, sysctl, tload, top, vmstat, w and watch

Descriptions

free

free displays the total amount of free and used physical and swap memory in the system, as well as the shared memory and buffers used by the kernel.

kill

kills sends signals to processes.

oldps and ps

ps gives a snapshot of the current processes.

pgrep

pgrep looks up processes based on name and other attributes.

pkill

pkill signals processes based on name and other attributes.

skill

skill sends signals to process matching a criteria.

snice

snice changes the scheduling priority for process matching a criteria.

sysctl

sysctl modifies kernel parameters at runtime.

tload

tload prints a graph of the current system load average to the specified tty (or the tty of the tload process if none is specified).

top

top provides an ongoing look at processor activity in real time.

vmstat

vmstat reports information about processes, memory, paging, block IO, traps, and cpu activity.

w

w displays information about the users currently on the machine, and their processes.

watch

watch runs command repeatedly, displaying its output (the first screen full).

Library Files

libproc.so

Descriptions

libproc

libproc is the library against which most of the programs in this set are linked to save disk space by implementing common functions only once.

Procps Installation Dependencies

Last checked against version 2.0.7.

Bash: sh Binutils: as, ld, strip Fileutils: install, ln, mv, rm Gcc: cc1, collect2, cpp0, gcc Grep: grep Make: make Gawk: awk Sed: sed Sh−utils: basename, pwd Textutils: sort, tr

Installing Psmisc−21

```
Estimated build time:           0.11 SBU
Estimated required disk space:  2 MB
```

Installation of Psmisc

Install Psmisc by running the following commands:

```
./configure --prefix=/usr --exec-prefix=/ &&
make &&
make install
```

Psmisc installs the `/usr/share/man/man1/pidof.1` man page, but Psmisc's pidof program isn't installed by default. Generally this isn't a problem, because we install the Sysvinit package later on which provides a better pidof program.

It's up to you now to decide if you are going to use the Sysvinit package which provides a pidof program, or not. If you are going to, you should remove Psmisc's pidof man page by running:

```
rm /usr/share/man/man1/pidof.1
```

If you're not going to use Sysvinit, you should complete this package's installation by creating the `/bin/pidof` symlink by running:

```
ln -s killall /bin/pidof
```

Command explanations

--exec-prefix=/: This will cause the programs to be installed in `/bin` rather than in `/usr/bin`. The programs in this package are often used in bootscripts, so they should be in the `/bin` directory so they can be used when the `/usr` partition isn't mounted yet.

Contents of Psmisc

Last checked against version 21.

Program Files

fuser, killall and pstree

Note that in LFS we don't install the pidof link by default because we use pidof from sysvinit instead.

Descriptions

fuser

fuser displays the PIDs of processes that use the specified files or file systems.

killall

killall sends a signal to all processes running any of the specified commands.

pstree

pstree shows running processes as a tree.

Psmisc Installation Dependencies

Last checked against version 20.2.

Autoconf: autoconf, autoheader Automake: aclocal, automake Bash: sh Bison: bison Binutils: as, ld Diffutils: cmp Fileutils: chmod, install, ls, mkdir, mv, rm Gettext: msgfmt, xgettext Gcc: cc, cc1, collect2, cpp0, gcc Grep: egrep, grep M4: m4 Make: make Gawk: gawk Sed: sed Sh−utils: basename, echo, expr, hostname, sleep, uname Texinfo: makeinfo Textutils: cat, tr

Installing Shadow−4.0.3

```
Estimated build time:           0.88 SBU
Estimated required disk space:  7 MB
```

Installation of Shadow Password Suite

Before you install this package, you may want to have a look at the Shadow hint. It discusses how you can make your system more secure regarding passwords, such as how to enable the more secure MD5 passwords, and how to get the most out of this Shadow package. The Shadow hint can be found at http://hints.linuxfromscratch.org/hints/shadowpasswd_plus.txt.

Install the Shadow Password Suite by running the following commands:

```
./configure --prefix=/usr \
    --libdir=/usr/lib --enable-shared &&
make &&
make install &&
cp etc/{limits,login.access} /etc &&
sed 's%/var/spool/mail%/var/mail%' \
    etc/login.defs.linux > /etc/login.defs &&
ln -s vipw /usr/sbin/vigr &&
rm /bin/vipw &&
mv /bin/sg /usr/bin &&
mv /usr/lib/lib{shadow,misc}.so.0* /lib &&
ln -sf ../../lib/libshadow.so.0 /usr/lib/libshadow.so &&
ln -sf ../../lib/libmisc.so.0 /usr/lib/libmisc.so
```

Sh−utils and Shadow Password Suite each install a unique `groups` program. If you wish, you may remove the `groups` program installed by the Shadow Password Suite by running the following command:

```
rm /bin/groups
```

Command explanations

`cp limits login.access /etc`: These files were not installed during the installation of the package so we copy them manually as those files are used to configure authentication details on the system.

`sed "s%/var/spool/mail%/var/mail%" login.defs.linux > /etc/login.defs`: /var/spool/mail is the old location of the user mailboxes. The location that is used nowadays is /var/mail.

`ln -s vipw vigr`: According to the manpage of vipw, vigr should be a symlink to it. Because the shadow installation procedure doesn't create these symlinks, we create them manually.

Contents of Shadow

Last checked against version 4.0.3.

Program Files

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

Descriptions

chage

chage changes the number of days between password changes and the date of the last password change.

chfn

chfn changes a user's full name and other information (office room number, office phone number, and home phone number).

chpasswd

chpasswd reads a file of user name and password pairs from standard input and uses this information to update a group of existing users.

chsh

chsh changes the user login shell.

dpasswd

dpasswd adds, deletes, and updates dial–up passwords for user login shells.

expiry

expiry checks and enforces a password expiration policy.

faillog

faillog formats the contents of the failure log,/var/log/faillog, and maintains failure counts and limits.

gpasswd

gpasswd is used to administer the /etc/group file.

groupadd

The groupadd command creates a new group account using the values specified on the command line and the default values from the system.

groupdel

The groupdel command modifies the system account files, deleting all entries that refer to group.

groupmod

The groupmod command modifies the system account files to reflect the changes that are specified on the command line.

groups

groups prints the groups which a user is in.

grpck

grpck verifies the integrity of the system authentication information.

grpconv

grpunconv converts to shadow group files from normal group files.

grpunconv

grpunconv converts from shadow group files to normal group files.

lastlog

lastlog formats and prints the contents of the last login log, /var/log/lastlog. The login−name, port, and last login time will be printed.

login

login is used to establish a new session with the system.

logoutd

logoutd enforces the login time and port restrictions specified in /etc/porttime.

mkpasswd

mkpasswd reads a file in the format given by the flags and converts it to the corresponding database file format.

newgrp

newgrp is used to change the current group ID during a login session.

newusers

newusers reads a file of user name and clear text password pairs and uses this information to update a group of existing users or to create new users.

passwd

passwd changes passwords for user and group accounts.

pwck

pwck verifies the integrity of the password files.

pwconv

pwconv converts the normal password file to a shadowed password file.

pwunconv

pwunconv converts a shadowed password file to a normal password file.

sg

sg sets the user's GID to that of the given group, or executes a given command as member of the given group.

useradd

useradd creates a new user or update default new user information.

userdel

userdel modifies the system account files, deleting all entries that refer to a specified login name.

usermod

usermod modifies the system account files to reflect the changes that are specified on the command line.

vipw and vigr

vipw and vigr will edit the files /etc/passwd and /etc/group, respectively. With the −s flag, they will edit the shadow versions of those files, /etc/shadow and /etc/gshadow, respectively.

Library Files

libmisc.[a,so], libshadow.[a,so]

Descriptions

libmisc

No description is currently available.

libshadow

libshadow provides common functionality for the shadow programs.

Shadow Installation Dependencies

Last checked against version 20001016.

Autoconf: autoconf, autoheader Automake: aclocal, automake Bash: sh Binutils: ar, as, ld, nm, ranlib Diffutils: cmp Fileutils: chmod, cp, install, ln, ls, mkdir, mv, rm, rmdir Gettext: msgfmt, xgettext Gcc: cc1, collect2, cpp0, gcc Glibc: ldconfig Grep: egrep, grep M4: m4 Make: make Gawk: gawk

Net−tools: hostname Sed: sed Sh−utils: basename, echo, expr, sleep, uname Texinfo: makeinfo Textutils: cat, sort, tr, uniq

Installing Sysklogd−1.4.1

```
Estimated build time:           0.03 SBU
Estimated required disk space:  472 KB
```

Installation of Sysklogd

Install Sysklogd by running the following commands:

```
make &&
make install
```

Contents of Sysklogd

Last checked against version 1.4.1.

Program Files

klogd and syslogd

Descriptions

klogd

klogd is a system daemon which intercepts and logs Linux kernel messages.

syslogd

syslogd provides the kind of logging that many modern programs use. Every logged message contains at least a time and a hostname field, normally a program name field, too, but that depends on how trusty the logging program is.

Sysklogd Installation Dependencies

Last checked against version 1.4.1.

Binutils: as, ld, strip Fileutils: install Gcc: cc1, collect2, cpp0, gcc Make: make

Installing Sysvinit−2.84

```
Estimated build time:           0.06 SBU
Estimated required disk space:  1 MB
```

Installation of Sysvinit

When run levels are changed (for example, when halting the system) init sends the TERM and KILL signals to the processes which it started. init prints "Sending processes the TERM signal" to the screen. This seems to imply that init is sending these signals to all the currently running processes. To avoid this confusion, the init.c file can be modified, so that the sentence reads "Sending processes started by init the TERM signal", by running the following commands. If you don't want to change it, skip it.

```
cp src/init.c{,.backup} &&
sed 's/Sending processes/Sending processes started by init/g' \
    src/init.c.backup > src/init.c
```

Install Sysvinit by running the following commands:

```
make −C src &&
make −C src install
```

Contents of Sysvinit

Last checked against version 2.84.

Program Files

halt, init, killall5, last, lastb (link to last), mesg, pidof (link to killall5), poweroff (link to halt), reboot (link to halt), runlevel, shutdown, sulogin, telinit (link to init), utmpdump and wall

Descriptions

halt

halt notes that the system is being brought down in the file /var/log/wtmp, and then either tells the kernel to halt, reboot or poweroff the system. If halt or reboot is called when the system is not in runlevel 0 or 6, shutdown will be invoked instead (with the flag −h or −r).

init

init is the parent of all processes. Its primary role is to create processes from a script stored in the file /etc/inittab. This file usually has entries which cause init to spawn gettys on each line that users can log in. It also controls autonomous processes required by any particular system.

killall5

killall5 is the SystemV killall command. It sends a signal to all processes except the processes in its own session, so it won't kill the shell that is running the script it was called from.

last

last searches back through the file /var/log/wtmp (or the file designated by the −f flag) and displays a list of all users logged in (and out) since that file was created.

lastb

lastb is the same as last, except that by default it shows a log of the file /var/log/btmp, which contains all the bad login attempts.

mesg

mesg controls the access to the user's terminal by others. It's typically used to allow or disallow other users to write to his terminal.

pidof

pidof displays the process identifiers (PIDs) of the named programs.

poweroff

poweroff is equivalent to shutdown −h −p now. It halts the computer and switches off the computer (when using an APM compliant BIOS and APM is enabled in the kernel).

reboot

reboot is equivalent to shutdown −r now. It reboots the computer.

runlevel

runlevel reads the system utmp file (typically /var/run/utmp) to locate the runlevel record, and then prints the previous and current system runlevel on its standard output, separated by a single space.

shutdown

shutdown brings the system down in a secure way. All logged−in users are notified that the system is going down, and login is blocked.

sulogin

sulogin is invoked by init when the system goes into single user mode (this is done through an entry in /etc/inittab). Init also tries to execute sulogin when it is passed the −b flag from the boot loader (LILO, for example).

telinit

telinit sends appropriate signals to init, telling it which runlevel to change to.

utmpdump

utmpdumps prints the content of a file (usually /var/run/utmp) on standard output in a user friendly format.

wall

wall sends a message to everybody logged in with their mesg permission set to yes.

Sysvinit Installation Dependencies

Last checked against version 2.84.

Bash: sh Binutils: as, ld Fileutils: chown, cp, install, ln, mknod, rm Gcc: cc, cc1, collect2, cpp0
Make: make Sed: sed

Installing Tar−1.13

```
Estimated build time:           0.26 SBU
Estimated required disk space:  6 MB
```

Installation of Tar

If you want to be able to directly use bzip2 files with tar, you can use the tar patch available from the LFS FTP site. This patch will add the −j option to tar which works the same as the −z option to tar (which can be used for gzip files).

Apply the patch by running the following command:

```
patch -Np1 -i ../tar-1.13.patch
```

Install Tar by running the following commands:

```
./configure --prefix=/usr \
    --libexecdir=/usr/bin --bindir=/bin &&
make &&
make install
```

Contents of Tar

Last checked against version 1.13.

Program Files

rmt and tar

Descriptions

rmt

rmt is a program used by the remote dump and restore programs in manipulating a magnetic tape drive through an interprocess communication connection.

tar

tar is an archiving program designed to store and extract files from an archive file known as a tar file.

Tar Installation Dependencies

Last checked against version 1.13.

Autoconf: autoconf, autoheader Automake: aclocal, automake Bash: sh Binutils: ar, as, ld, ranlib
Diffutils: cmp Fileutils: chmod, install, ls, mv, rm Gettext: msgfmt, xgettext
Gcc: cc, cc1, collect2, cpp0, gcc Glibc: getconf Grep: egrep, fgrep, grep M4: m4 Make: make
Gawk: gawk Net−tools: hostname Patch: patch Sed: sed Sh−utils: basename, echo, expr, sleep, uname
Texinfo: install−info, makeinfo Textutils: cat, tr

Installing Util−linux−2.11u

```
Estimated build time:           0.38 SBU
Estimated required disk space:  10 MB
```

FHS compliance notes

The FHS recommends that we use /var/lib/hwclock as the location of the adjtime file, instead of the usual /etc. To make hwclock, which is part of the util−linux package, FHS−compliant, run the following.

```
cp hwclock/hwclock.c{,.backup} &&
sed 's%etc/adjtime%var/lib/hwclock/adjtime%' \
    hwclock/hwclock.c.backup > hwclock/hwclock.c &&
mkdir -p /var/lib/hwclock
```

Installation of Util−linux

Install Util−linux by running the following commands:

```
./configure &&
make HAVE_SLN=yes &&
make HAVE_SLN=yes install
```

Command explanations

HAVE_SLN=yes: We don't build this program because it already was installed by Glibc.

Contents of Util−linux

Last checked against version 2.11t.

Program Files

agetty, arch, blockdev, cal, cfdisk, chkdupexe, col, colcrt, colrm, column, ctrlaltdel, cytune, ddate, dmesg, elvtune, fdformat, fdisk, fsck.cramfs, fsck.minix, getopt, hexdump, hwclock, ipcrm, ipcs, isosize, line, logger, look, losetup, mcookie, mkfs, mkfs.bfs, mkfs.cramfs, mkfs.minix, mkswap, more, mount, namei, parse.bash, parse.tcsh, pg, pivot_root, ramsize (link to rdev), raw, rdev, readprofile, rename, renice, rev, rootflags (link to rdev), script, setfdprm, setsid, setterm, sfdisk, swapoff (link to swapon), swapon, test.bash, test.tcsh, tunelp, ul, umount, vidmode (link to rdev), whereis and write

Descriptions

agetty

agetty opens a tty port, prompts for a login name and invokes the /bin/login command.

arch

arch prints the machine architecture.

blockdev

blockdev allows to call block device ioctls from the command line.

cal

cal displays a simple calender.

cfdisk

cfdisk is a libncurses based disk partition table manipulator.

chkdupexe

chkdupexe finds duplicate executables.

col

col filters reverse line feeds from input.

colcrt

colcrt filters nroff output for CRT previewing.

colrm

colrm removes columns from a file.

column

column columnates lists.

ctrlaltdel

ctrlaltdel sets the function of the CTRL+ALT+DEL key combination (hard or soft reset).

cytune

cytune queries and modifies the interruption threshold for the Cyclades driver.

ddate

ddate converts Gregorian dates to Discordian dates.

dmesg

dmesg is used to examine or control the kernel ring buffer (boot messages from the kernel).

elvtune

elvtune allows to tune the I/O elevator per block device queue basis.

fdformat

fdformat low−level formats a floppy disk.

fdisk

fdisk is a disk partition table manipulator.

fsck.cramfs

No description is currently available.

fsck.minix

fsck.minix performs a consistency check for the Linux MINIX filesystem.

getopt

getops parses command options the same way as the getopt C command.

hexdump

hexdump displays specified files, or standard input, in a user specified format (ascii, decimal, hexadecimal, octal).

hwclock

hwclock queries and sets the hardware clock (Also called the RTC or BIOS clock).

ipcrm

ipcrm removes a specified resource.

ipcs

ipcs provides information on IPC facilities.

isosize

isosize outputs the length of a iso9660 file system.

line

line copies one line (up to a newline) from standard input and writes it to standard output.

logger

logger makes entries in the system log.

look

look displays lines beginning with a given string.

losetup

losetup sets up and controls loop devices.

mcookie

mcookie generates magic cookies for xauth.

mkfs

mkfs builds a Linux filesystem on a device, usually a harddisk partition.

mkfs.bfs

mkfs.bfs creates a SCO bfs file system on a device, usually a harddisk partition.

mkfs.cramfs

No description is currently available.

mkfs.minix

mkfs.minix creates a Linux MINIX filesystem on a device, usually a harddisk partition.

mkswap

mkswap sets up a Linux swap area on a device or in a file.

more

more is a filter for paging through text one screen full at a time.

mount

mount mounts a filesystem from a device to a directory (mount point).

namei

namei follows a pathname until a terminal point is found.

parse.bash, parse.tcsh, test.bash, test.tcsh

These are example scripts for using the getopt program with either BASH or TCSH.

pg

No description is currently available.

pivot_root

pivot_root moves the root file system of the current process.

ramsize

ramsize queries and sets RAM disk size.

raw

raw is used to bind a Linux raw character device to a block device.

rdev

rdev queries and sets image root device, swap device, RAM disk size, or video mode.

readprofile

readprofile reads kernel profiling information.

rename

rename renames files.

renice

renice alters priority of running processes.

rev

rev reverses lines of a file.

rootflags

rootflags queries and sets extra information used when mounting root.

script

script makes typescript of terminal session.

setfdprm

setfdprm sets user–provides floppy disk parameters.

setsid

setsid runs programs in a new session.

setterm

setterm sets terminal attributes.

sfdisk

sfdisk is a disk partition table manipulator.

swapoff

swapoff disables devices and files for paging and swapping.

swapon

swapon enables devices and files for paging and swapping.

tunelp

tunelp sets various parameters for the LP device.

ul

ul reads a file and translates occurrences of underscores to the sequence which indicates underlining for the terminal in use.

umount

umount unmounts a mounted filesystem.

vidmode

vidmode queries and sets the video mode.

whereis

whereis locates a binary, source and manual page for a command.

write

write sends a message to another user.

Util−linux Installation Dependencies

Last checked against version 2.11n.

Bash: sh Binutils: as, ld Diffutils: cmp Fileutils: chgrp, chmod, cp, install, ln, mv, rm
Gettext: msgfmt, xgettext Gcc: cc, cc1, collect2, cpp, cpp0 Glibc: rpcgen Grep: grep Make: make
Sed: sed Sh−utils: uname, whoami Textutils: cat

Installing LFS−Bootscripts−1.10

```
Estimated build time:           0.01 SBU
Estimated required disk space:  420 KB
```

Installation of LFS−Bootscripts

We will be using SysV style init scripts. We have chosen this style because it is widely used and we feel comfortable with it. If you would prefer to try something else, Marc Heerdink has written a hint about BSD style init scripts, which may be found at http://hints.linuxfromscratch.org/hints/bsd−init.txt.

If you decide to use BSD style, or some other style scripts, you can skip Chapter 7 when you arrive at it and move on to Chapter 8.

Install LFS−Bootscripts by running the following command:

```
cp -a rc.d sysconfig /etc &&
chown -R root:root /etc/rc.d /etc/sysconfig
```

Contents of LFS−bootscripts

Last checked against version 1.10.

Scripts

checkfs, cleanfs, functions, halt, ifdown, ifup, loadkeys, localnet, mountfs, mountproc, network, rc, reboot, sendsignals, setclock, swap, sysklogd and template

Descriptions

checkfs

The checkfs script checks the file systems just before they are mounted (with the exception of journal and network based file systems).

cleanfs

The cleanfs script removes files that shouldn't be preserved between reboots, such as /var/run/*, /var/lock/*, it re−creates /var/run/utmp and removes the possible present /etc/nologin, /fastboot and /forcefsck files.

functions

The functions script contains shared functions among different scripts such as error checking, status checking, etc.

halt

The halt script halts the system.

ifdown, ifup

The ifdown and ifup scripts assist the network script with network devices.

loadkeys

The loadkeys script loads the proper keymap table that matches your keyboard layout.

localnet

The localnet script sets up the system's hostname and local loopback device.

mountfs

The mountfs script mounts all file systems that aren't marked noauto or aren't network based.

mountproc

The mountproc script is used to mount the proc filesystem.

network

The network script sets up network interfaces (such as network cards) and sets up the default gateway where applicable.

rc

The rc script is the master runlevel control script which is responsible for running all the other scripts one−by−one in a specific sequence.

reboot

The reboot scripts reboots the system.

sendsignals

The sendsignals script makes sure every process is terminated before the system reboots or halts.

setclock

The setclock scripts resets the kernel clock to localtime in case the hardware clock isn't set to GMT time.

swap

The swap scripts enables and disables swap files and partitions.

sysklogd

The sysklogd script start and stops the system and kernel log daemons.

template

The template script is a template you can use to create your own bootscripts for your other daemons.

LFS−Bootscripts Installation Dependencies

Last checked against version 1.10.

Fileutils: chown, cp

Configuring essential software

Now that all software is installed, all that we need to do to get a few programs running properly is to create their configuration files.

Configuring Vim

By default vim runs in vi compatible mode. Some people might like this, but we have a high preference to run vim in vim mode (else we wouldn't have included vim in this book, but the original vi). Create the `/root/.vimrc` by running the following:

```
cat > /root/.vimrc << "EOF"
" Begin /root/.vimrc

set nocompatible
set bs=2

" End /root/.vimrc
EOF
```

Configuring Glibc

We need to create the /etc/nsswitch.conf file. Although glibc should provide defaults when this file is missing or corrupt, its defaults don't work well with networking which will be dealt with in a later chapter. Also, our timezone needs to be set up.

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

publickey: files

hosts: files dns
networks: files

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

netgroup: db files

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

The **`tzselect`** script has to be run and the questions regarding your timezone have to be answered. When you're done, the script will give the location of the needed timezone file.

Create the `/etc/localtime` symlink by running:

```
ln -sf ../usr/share/zoneinfo/<tzselect's output> /etc/localtime
```

tzselect's output can be something like *EST5EDT* or *Canada/Eastern*.

The symlink you'd create with that information would be:

```
ln -sf ../usr/share/zoneinfo/EST5EDT /etc/localtime
```

Or:

```
ln -sf ../usr/share/zoneinfo/Canada/Eastern /etc/localtime
```

Configuring Dynamic Loader

By default, the dynamic loader (`/lib/ld-linux.so.2`) searches through `/lib` and `/usr/lib` for dynamic libraries that are needed by programs when you run them. However, if there are libraries in directories other than `/lib` and `/usr/lib`, you need to add them 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 we 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
```

Configuring Sysklogd

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

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

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

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

Configuring Shadow Password Suite

This package contains utilities to modify users's passwords, add or delete users and groups, and the like. We're not going to explain what 'password shadowing' means. A full explanation can be found in the doc/HOWTO file within the unpacked shadow password suite's source tree. There's one thing to keep in mind if you decide to use shadow support: that programs that need to verify passwords (for example xdm, ftp daemons, pop3 daemons) need to be 'shadow−compliant', that is they need to be able to work with shadow'ed passwords.

To enable shadow'ed passwords, run the following command:

```
/usr/sbin/pwconv
```

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

1:2345:respawn:/sbin/agetty tty1 9600
2:2345:respawn:/sbin/agetty tty2 9600
3:2345:respawn:/sbin/agetty tty3 9600
4:2345:respawn:/sbin/agetty tty4 9600
5:2345:respawn:/sbin/agetty tty5 9600
6:2345:respawn:/sbin/agetty tty6 9600

# End /etc/inittab
EOF
```

Configuring your keyboard

Nothing is more annoying than using Linux with a wrong keymap loaded for your keyboard. If you have a default US keyboard, you can skip this section. The US keymap file is the default if you don't change it.

To set the default keymap file, create the `/usr/share/kbd/keymaps/defkeymap.map.gz` symlink by running the following commands:

```
ln -s <path/to/keymap> /usr/share/kbd/keymaps/defkeymap.map.gz
```

Replace <path/to/keymap> with the your keyboard's map file. For example, if you have a Dutch keyboard, you would run:

```
ln -s i386/qwerty/nl.map.gz /usr/share/kbd/keymaps/defkeymap.map.gz
```

A second option to configure your keyboard's layout is to compile the keymap directly into the kernel. This will make sure that your keyboard always works as expected, even when you have booted into maintenance mode (by passing `init=/bin/sh' to the kernel) in which case the bootscript that normally sets up your keymap isn't run.

Run the following command to patch the correct keymap into the kernel source. You will have to repeat this command whenever you unpack a new kernel:

```
loadkeys -m /usr/share/kbd/keymaps/defkeymap.map.gz > \
```

```
    /usr/src/linux/drivers/char/defkeymap.c
```

Creating the /var/run/utmp, /var/log/wtmp and /var/log/btmp files

Programs like login, shutdown, uptime and others want to read from and write to the /var/run/utmp /var/log/btmp and /var/log/wtmp. These files contain information about who is currently logged in. It also contains information on when the computer was last booted and shutdown and a record of the bad login attempts.

Create these files with their proper permissions by running the following commands:

```
touch /var/run/utmp /var/log/{btmp,lastlog,wtmp} &&
chmod 644 /var/run/utmp /var/log/{btmp,lastlog,wtmp}
```

Creating root password

Choose a password for user root and create it by running the following command:

```
passwd root
```

Chapter 7. Setting up system boot scripts

Introduction

This chapter will set up the bootscripts that you installed in chapter 6. Most of these scripts will work without needing to modify them, but a few do require additional configuration files set up as they deal with hardware dependent information.

How does the booting process with these scripts work?

Linux uses a special booting facility named SysVinit. It's based on a concept of *runlevels*. It can be widely different from one system to another, so it can't be assumed that because things worked in <insert distro name> they should work like that in LFS too. LFS has its own way of doing things, but it respects generally accepted standards.

SysVinit (which we'll call *init* from now on) works using a runlevels scheme. There are 7 (from 0 to 6) runlevels (actually, there are more runlevels but they are for special cases and generally not used. The init man page describes those details), and each one of those corresponds to the things the computer is supposed to do when it starts up. The default runlevel is 3. Here are the descriptions of the different runlevels as they are often 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 runlevels is **`init <runlevel>`** where <runlevel> is the target runlevel. For example, to reboot the computer, a user would issue the init 6 command. The reboot command is just an alias, as is the halt command an alias to init 0.

There are a number of directories under /etc/rc.d that look like like rc?.d where ? is the number of the runlevel and rcsysinit.d which contain 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 sooner it gets executed. When init switches to another runlevel, the appropriate services get killed and others get started.

The real scripts are in /etc/rc.d/init.d. They do all the work, and the symlinks all point to them. Killing links and starting links point to the same script in /etc/rc.d/init.d. That's because the scripts can be called with different parameters like start, stop, restart, reload, status. When a K link is encountered, the appropriate script is run with the stop argument. When a S link is encountered, the appropriate script is run with the start argument.

There is one exception. 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 it is that when you are going to reboot or halt the system, you don't want to start anything, only stop the system.

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 doesn't 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's your own LFS system). The files given here are just an example of how it can be done in a nice way (well, what we consider nice –– you may hate it).

Configuring the setclock script

This setclock script reads the time from your hardware clock (also known as BIOS or CMOS clock) and either converts that time to localtime using the `/etc/localtime` file (if the hardware clock is set to GMT) or not (if the hardware clock is already set to localtime). There is no way to auto−detect whether the hardware clock is set to GMT or not, so we need to configure that here ourselves.

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

Now, you may want to take a look at a very good hint explaining how we deal with time on LFS at http://hints.linuxfromscratch.org/hints/time.txt. It explains issues such as timezones, UTC, and the TZ environment variable.

Do I need the loadkeys script?

If you decided to compile your keymap file directly into the kernel back at the end of Chapter 6, then you strictly speaking don't need to run this loadkeys script, since the kernel has already set up the keymap for you. You can still run it if you want, it isn't going to hurt you. It could even be beneficial to keep it in case you run a lot of different kernels and don't remember or want to compile the keymap into every kernel you lay your hands on.

If you decided you don't need to, or don't want to use the loadkeys script, remove the `/etc/rc.d/rcsysinit.d/S70loadkeys` symlink.

Configuring the sysklogd script

The `sysklogd` script invokes the **`syslogd`** program with the *−m 0* option. This option turns off the periodic timestamp mark that syslogd writes to the log files every 20 minutes by default. If you want to turn on this periodic timestamp mark, edit the `sysklogd` script and make the changes accordingly. See **`man syslogd`** for more information.

Configuring the localnet script

Part of the localnet script is setting up the system's hostname. This needs to be configured in the /etc/sysconfig/network.

Create the /etc/sysconfig/network file and enter a hostname by running:

```
echo "HOSTNAME=lfs" > /etc/sysconfig/network
```

"lfs" needs to be replaced with the name the computer is to be called. You should not enter the FQDN (Fully Qualified Domain Name) here. That information will be put in the `/etc/hosts` file later on.

Creating the /etc/hosts file

If a network card is to be configured, you have to decide on the IP−address, FQDN and possible aliases for use in the /etc/hosts file. The syntax is:

```
<IP address> myhost.mydomain.org aliases
```

You should made sure that the IP−address is in the private network IP−address range. Valid ranges are:

```
        Class Networks
        A     10.0.0.0
        B     172.16.0.0 through 172.31.0.0
        C     192.168.0.0 through 192.168.255.0
```

A valid IP address could be 192.168.1.1. A valid FQDN for this IP could be www.linuxfromscratch.org.

If you aren't going to use a network card, you still need to come up with a FQDN. This is necessary for certain programs to operate correctly.

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 www.mydomain.com <value of HOSTNAME> localhost

# End /etc/hosts (no network card version)
EOF
```

If a network card is to be configured, create the `/etc/hosts` file by running:

```
cat > /etc/hosts << "EOF"
# Begin /etc/hosts (network card version)

127.0.0.1 localhost.localdomain localhost
192.168.1.1 www.mydomain.org <value of HOSTNAME>

# End /etc/hosts (network card version)
EOF
```

Of course, the 192.168.1.1 and www.mydomain.org have to be changed to your liking (or requirements if assigned an IP−address by a network/system administrator and this machine is planned to be connected to an existing network).

Configuring the network script

This section only applies if you're going to configure a network card.

If you don't have any network cards, you are most likely not going to create any configuration files relating to network cards. If that is the case, you must remove the `network` symlinks from all the runlevel directories (`/etc/rc.d/rc*.d`)

Configuring default gateway

If you're on a network you may need to set up the default gateway for this machine. This is done by adding the proper values to the /etc/sysconfig/network file by running the following:

```
cat >> /etc/sysconfig/network << "EOF"
GATEWAY=192.168.1.2
GATEWAY_IF=eth0
EOF
```

The values for GATEWAY and GATEWAY_IF need to be changed to match your network setup. GATEWAY contains the IP address of the default gateway, and GATEWAY_IF contains the network interface through which the default gateway can be reached.

Creating network interface configuration files

Which interfaces are brought up and down by the network script depends on the files in the /etc/sysconfig/network−devices directory. This directory should contain files in the form of ifconfig.xyz, where xyz is a network interface name (such as eth0 or eth0:1)

If you decide to rename or move this /etc/sysconfig/network−devices directory, make sure you update the /etc/sysconfig/rc file as well and update the network_devices by providing it with the new path.

Now, new files are created in that directory containing the following. The following command creates a sample ifconfig.eth0 file:

```
cat > /etc/sysconfig/network-devices/ifconfig.eth0 << "EOF"
ONBOOT=yes
IP=192.168.1.1
NETMASK=255.255.255.0
BROADCAST=192.168.1.255
EOF
```

Of course, the values of those variables have to be changed in every file to match the proper setup. If the ONBOOT variable is set to yes, the network script will bring it up during the booting of the system. If set to anything else but yes, it will be ignored by the network script and thus not brought up.

Chapter 8. Making the LFS system bootable

Introduction

This chapter will make LFS bootable. This chapter deals with creating a new fstab file, building a new kernel for the new LFS system and adding the proper entries to LILO so that the LFS system can be selected for booting at the LILO: prompt.

Creating the /etc/fstab file

In order for certain programs to be able to determine where certain partitions are supposed to be mounted by default, the /etc/fstab file is used. Create a new file `/etc/fstab` containing the following:

```
cat > /etc/fstab << "EOF"
# Begin /etc/fstab

# filesystem   mount-point fs-type    options     dump    fsck-order

/dev/*LFS*     /           *fs-type*  defaults    1       1
/dev/*swap*    swap        swap       pri=1       0       0
proc           /proc       proc       defaults    0       0

# End /etc/fstab
EOF
```

LFS, ***swap*** and ***fs-type*** have to be replaced with the appropriate values (/dev/hda2, /dev/hda5 and reiserfs for example).

When adding a reiserfs partition, the **1 1** at the end of the line should be replaced with **0 0**.

For more information on the various fields which are in the fstab file, see **man 5 fstab**.

There are other lines which you may consider adding to your fstab file. One example is the line which you must have if you are using devpts:

```
devpts         /dev/pts    devpts     gid=4,mode=620 0    0
```

Another example is a line to use if you intend to use USB devices:

```
usbdevfs       /proc/bus/usb usbdevfs defaults    0       0
```

Both of these options will only work if you have the relevant support compiled into your kernel.

Installing Linux−2.4.19

```
Estimated build time:           All default options: 4.20 SBU
Estimated required disk space:  All default options: 181 MB
```

Building the kernel involves a few steps: configuring it and compiling it. There are a few ways to configure the kernel. If you don't like the way this book does it, read the `README` that comes with the kernel source tree, and find out what the other options are.

Something you could do, is take the `.config` file from your host distribution's kernel source tree and copy it to `$LFS/usr/src/linux-2.4.19`. This way you don't have to configure the entire kernel from scratch and can use your current values. If you choose to do this, first run the **`make mrproper`** command below, then copy the `.config` file over, then run **`make menuconfig`** followed by the rest of the commands (**`make oldconfig`** may be better in some situations. See the `README` file for more details when to use **`make oldconfig`**).

If you intend to use kernel modules, you will need an `/etc/modules.conf` file. Information pertaining to modules and to kernel configuration in general may be found in the kernel documentation, which is stored stored in `/usr/src/linux-2.4.19/Documentation`. The modules.conf man page and the kernel HOWTO at http://www.tldp.org/HOWTO/Kernel−HOWTO.html may also be of interest to you.

The following commands are run to build the kernel:

```
make mrproper &&
make menuconfig &&
make dep &&
make bzImage &&
make modules &&
make modules_install &&
cp arch/i386/boot/bzImage /boot/lfskernel &&
cp System.map /boot
```

Note: the arch/i386/boot/bzImage path may vary on different platforms.

Linux Installation Dependencies

Last checked against version 2.4.17.

Bash: sh Binutils: ar, as, ld, nm, objcopy Fileutils: cp, ln, mkdir, mv, rm, touch
Findutils: find, xargs Gcc: cc1, collect2, cpp0, gcc Grep: grep Gzip: gzip Make: make Gawk: awk
Modutils: depmod, genksyms Net−tools: dnsdomainname, hostname Sed: sed
Sh−utils: basename, date, expr, pwd, stty, uname, whoami, yes Textutils: cat, md5sum, sort, tail

Making the LFS system bootable

In order to be able to boot the LFS system, we need to update our bootloader. We're assuming that your host system is using Lilo (since that's the most commonly used boot loader at the moment).

We will not be running the lilo program inside chroot. Running lilo inside chroot can have fatal side−effects which render your MBR useless and you'd need a boot disk to be able to start any Linux system (either the host system or the LFS system).

First we'll exit chroot and copy the lfskernel file to the host system:

```
logout
cp $LFS/boot/lfskernel /boot
```

The next step is adding an entry to /etc/lilo.conf so that we can choose LFS when booting the computer:

```
cat >> /etc/lilo.conf << "EOF"
image=/boot/lfskernel
        label=lfs
        root=<partition>
        read-only
EOF
```

<partition> must be replaced with the LFS partition's designation.

Also note that if you are using reiserfs for your root partition, the line **`read−only`** should be changed to **`read−write`**.

Now, update the boot loader by running:

```
/sbin/lilo -v
```

The last step is synchronizing the host system's lilo configuration files with the LFS system's:

```
cp /etc/lilo.conf $LFS/etc &&
cp $(grep "image.*=" /etc/lilo.conf | cut -f 2 -d "=") $LFS/boot
```

Chapter 9. The End

The End

Well done! You have finished installing your LFS system. It may have been a long process, but we hope it was worth it. We wish you a lot of fun with your new shiny custom built Linux system.

Now would be a good time to strip all debug symbols from the binaries on your LFS system. If you are not a programmer and don't plan on debugging your software, then you will be happy to know that you can reclaim a few tens of megs by removing debug symbols. This process causes no inconvenience other than not being able to debug the software fully anymore, which is not an issue if you don't know how to debug.

Disclaimer: 98% of the people who use the command mentioned below don't experience any problems. But do make a backup of your LFS system before you run this command. There's a slight chance it may backfire on you and render your system unusable (mostly by destroying your kernel modules and dynamic & shared libraries). This is caused more often by typos than by a problem with the command used.

Having said that, the −−strip−debug option we use to strip is quite harmless under normal circumstances. It doesn't strip anything vital from the files. It also is quite safe to use −−strip−all on regular programs (don't use that on libraries − they will be destroyed), but it's not as safe, and the space you gain is not all that much. But if you're tight on disk space every little bit helps, so decide for yourself. Please refer to the strip man page for other strip options you can use. The general idea is to not run strip on libraries (other than −−strip−debug), just to be on the safe side.

```
find $LFS/{,usr/,usr/local/}{bin,sbin,lib} -type f \
   -exec /usr/bin/strip --strip-debug '{}' ';'
```

It may be a good idea to create the $LFS/etc/lfs file. By having this file it is very easy for you (and for us if you are going to ask for help with something at some point) to find out which LFS version you have installed on your system. Create the $LFS/etc/lfs file by running the following command:

```
echo 4.0-RC1 > $LFS/etc/lfs
```

Get Counted

Want to be counted as an LFS user now that you have finished the book? Head over to http://linuxfromscratch.org/cgi−bin/lfscounter.cgi and register as an LFS user by entering your name and the first LFS version you have used.

Let's reboot into LFS now...

Rebooting the system

Now that all software has been installed, bootscripts have been created, it's time to reboot the computer. Before we reboot let's unmount $LFS/proc and the LFS partition itself by running:

```
umount $LFS/proc &&
umount $LFS
```

If you decided to create multiple partitions, you need to umount the other partitions before you umount $LFS, like this:

```
umount $LFS/proc &&
umount $LFS/usr &&
umount $LFS/home &&
umount $LFS
```

And you can reboot your system by running something like:

```
/sbin/shutdown -r now
```

At the LILO: prompt make sure that you tell it to boot *lfs* and not the default entry which will boot your host system again.

After you have rebooted, your LFS system is ready for use and you can start adding your own software.

One final thing you may want to do is run lilo, now that you are booted into LFS. This way you will put the LFS version of LILO in the MBR rather than the one that's there right now from your host system. Depending on how old your host distribution is, the LFS version may have more advanced features you need/could use.

Either way, run the following to make the lilo version installed on LFS active:

```
/sbin/lilo
```

You may now remove the static directory. If you think you may need to redo Chapter 5, then you may wish to backup the directory before removing it. To remove the static directory, type the following command:

```
rm -rf /static
```

Now that you have finished installing your LFS system, you may be wondering how to install additional software, such as a web browser. Your first stop should be the Beyond Linux From Scratch project, which may be found at http://beyond.linuxfromscratch.org/. The LFS hints may also prove helpful, and are located at http://hints.linuxfromscratch.org/hints.shtml. On a similar note, if you are not only a newbie to LFS, but also to Linux in general, you may find the newbie hint at http://hints.linuxfromscratch.org/hints/newbie.txt very interesting.

Remember that there are several LFS mailinglists you may subscribe to if you are in need of help. See Chapter 1 − Mailing lists and archives for more information.

Again, we thank you for using the LFS Book and hope you found this book useful and worth your time.

III. Part III – Appendixes

Table of Contents

Appendix A. Package descriptions and dependencies

Introduction

In this appendix the following aspects of every package installed in this book are described:

- the official download location for the package,
- what the package contains,
- what each program from the package does,
- what the package needs to be compiled.

Most information about these packages (especially the descriptions of them) come from the man pages of those packages. We do not include the entire man page, but just some key elements to make it possible to understand what a program does. To get information on all details of a program, please refer to its man page or info page.

Certain packages are documented in more depth than others, because we just happen to know more about certain packages than about others. If you think anything should be added to the following descriptions, please don't hesitate to email the mailing lists. We intend that the list should contain an in−depth description of every package installed, but we can't do it without help.

Please note that currently only what a package does is described and not why it needs to be installed. This may be added later.

Also listed are all of the installation dependencies for all the packages that are installed in this book. The listings will include which programs from which packages are needed to successfully compile the package to be installed.

These are not running dependencies, meaning they don't tell you what programs are needed to use that packages programs. Just the ones needed to compile it.

The dependency list can be, from time to time, outdated in regards to the currently used package version. Checking dependencies takes quite a bit of work, so they may lag behind a bit on the package update. But often with minor package updates, the installation dependencies hardly change, so they'll be current in most cases. When we upgrade to a major new release, we'll make sure the dependencies are checked too.

Autoconf

Official Download Location

Autoconf (2.53): ftp://ftp.gnu.org/gnu/autoconf/

Contents of Autoconf

Last checked against version 2.53.

Program Files

autoconf, autoheader, autom4te, autoreconf, autoscan, autoupdate and ifnames

Descriptions

autoconf

autoconf is a tool for producing shell scripts that automatically configure software source code packages to adapt to many kinds of Unix−like systems. The configuration scripts produced by autoconf are independent of autoconf when they are run, so their users do not need to have autoconf.

autoheader

The autoheader program can create a template file of C #define statements for configure to use.

autom4te

autom4te runs GNU M4 on files.

autoreconf

If there are a lot of autoconf−generated configure scripts, the autoreconf program can save some work. It runs autoconf and autoheader (where appropriate) repeatedly to remake the autoconf configure scripts and configuration header templates in the directory tree rooted at the current directory.

autoscan

The autoscan program can help to create a configure.in file for a software package. autoscan examines the source files in a directory tree. If a directory is not specified on the command line, then the current working directory is used. The source files are searched for common portability problems and a configure.scan file is created to serve as the preliminary configure.in for that package.

autoupdate

The autoupdate program updates a configure.in file that calls autoconf macros by their old names to use the current macro names.

ifnames

ifnames can help when writing a configure.in for a software package. It prints the identifiers that the package already uses in C preprocessor conditionals. If a package has already been set up to have some portability, this program can help to determine what configure needs to check. It may fill in some gaps in a configure.in file generated by autoscan.

Autoconf Installation Dependencies

Last checked against version 2.52.

Bash: sh Diffutils: cmp Fileutils: chmod, install, ln, ls, mkdir, mv, rm Grep: fgrep, grep M4: m4
Make: make Gawk: gawk Sed: sed Sh−utils: echo, expr, hostname, sleep, uname Texinfo: install−info
Textutils: cat, tr

Automake

Official Download Location

Automake (1.6.3): ftp://ftp.gnu.org/gnu/automake/

Contents of Automake

Last checked against version 1.6.2.

Program Files

acinstall, aclocal, aclocal−1.6, automake, automake−1.6, compile, config.guess, config.sub, depcomp, elisp−comp, install−sh, mdate−sh, missing, mkinstalldirs, py−compile, ylwrap

Descriptions

acinstall

acinstall is a script which installs aclocal−style M4 files.

aclocal, aclocal−1.6

automake includes a number of autoconf macros which can be used in packages, some of which are needed by automake in certain situations. These macros must be defined in the aclocal.m4−file or they will not be seen by autoconf.

The aclocal program will automatically generate aclocal.m4 files based on the contents of configure.in. This provides a convenient way to get automake−provided macros without having to search around. Also, the aclocal mechanism is extensible for use by other packages.

automake, automake−1.6

To create all the Makefile.in's for a package, run the automake program in the top level directory, with no arguments. automake will automatically find each appropriate Makefile.am (by scanning configure.in) and generate the corresponding Makefile.in.

compile

compile is script which acts as a wrapper for compilers.

config.guess

config.guess is a script which attempts to guess a canonical system name.

config.sub

config.sub is a configuration validation subroutine script.

depcomp

depcomp is a script which compiles a program while generating dependencies as side−effects.

elisp−comp

elisp−comp is a script which byte−compiles .el files.

install−sh

install−sh is a script which installs a program, script, or a datafile.

mdate−sh

mdate−sh is a script which prints the modification time of a file or directory.

missing

missing is a script which acts as a common stub for a few missing GNU programs during an installation.

mkinstalldirs

mkinstalldirs is a script which makes a directory hierarchy.

py−compile

py−compile is a script which compiles a Python program.

ylwrap

ylwrap is a script which acts as a wrapper for lex/yacc invocations.

Automake Installation Dependencies

Last checked against version 1.5.

Bash: sh Diffutils: cmp Fileutils: chmod, install, ls, mkdir, mv, rm, rmdir Grep: fgrep, grep Make: make Perl: perl Sed: sed Sh−utils: echo, expr, hostname, sleep Texinfo: install−info Textutils: cat, tr

Bash

Official Download Location

Bash (2.05a): ftp://ftp.gnu.org/gnu/bash/

Contents of Bash

Last checked against version 2.05a.

Program Files

bash, sh (link to bash) and bashbug

Descriptions

bash

bash is the Bourne−Again SHell, which is a widely used command interpreter on Unix systems. The bash program reads from standard input, the keyboard. A user types something and the program will evaluate what he has typed and do something with it, like running a program.

bashbug

bashbug is a shell script to help the user compose and mail bug reports concerning bash in a standard format.

sh

sh is 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.

Bash Installation Dependencies

Last checked against version 2.05a.

Bash: bash, sh Binutils: ar, as, ld, ranlib, size Diffutils: cmp
Fileutils: chmod, cp, install, ln, ls, mkdir, mv, rm Gcc: cc, cc1, collect2, cpp0, gcc
Grep: egrep, grep Make: make Gawk: awk Sed: sed
Sh−utils: basename, echo, expr, hostname, sleep, uname Texinfo: install−info Textutils: cat, tr, uniq

Bin86

Official Download Location

Bin86 (0.16.3): http://www.cix.co.uk/~mayday/

Contents of Bin86

Last checked against version 0.16.3

Program Files

as86, as86_encap, ld86, nm86 (link to objdump86), objdump86 and size86 (link to objdump86)

Descriptions

as86

as86 is an assembler for the 8086...80386 processors.

as86_encap

as86_encap is a shell script to call as86 and convert the created binary into a C file prog.v to be included in or linked with programs like boot block installers.

ld86

ld86 understands only the object files produced by the as86 assembler. It can link them into either an impure or a separate I&D executable.

nm86

The symbol table of the binary file.

objdump86

Dumps detailed information about a binary file.

size86

Summary sizes of the data in a binary file.

Bin86 Installation Dependencies

Last checked against version 0.16.0.

Bash: sh Binutils: as, ld, strip Fileutils: chmod, install, ln, mv Gcc: cc, cc1, collect2, cpp0
Make: make Sed: sed

Binutils

Official Download Location

Binutils (2.13): ftp://ftp.gnu.org/gnu/binutils/

Contents of Binutils

Last checked against version 2.12.1.

Program Files

addr2line, ar, as, gasp, gprof, ld, nm, objcopy, objdump, ranlib, readelf, size, strings and strip

Descriptions

addr2line

addr2line translates program addresses into file names and line numbers. Given an address and an executable, it uses the debugging information in the executable to figure out which file name and line number are associated with a given address.

ar

The ar program creates, modifies, and extracts from archives. An archive is a single file holding a collection of other files in a structure that makes it possible to retrieve the original individual files (called members of the archive).

as

as is primarily intended to assemble the output of the GNU C compiler, gcc, for use by the linker ld.

gasp

gasp is the Assembler Macro Preprocessor.

gprof

gprof displays call graph profile data.

ld

ld combines a number of object and archive files, relocates their data and ties up symbol references. Often the last step in building a new compiled program to run is a call to ld.

nm

nm lists the symbols from object files.

objcopy

objcopy utility copies the contents of an object file to another. objcopy uses the GNU BFD Library to read and write the object files. It can write the destination object file in a format different from that of the source object file.

objdump

objdump displays information about one or more object files. The options control what particular information to display. This information is mostly useful to programmers who are working on the compilation tools, as opposed to programmers who just want their program to compile and work.

ranlib

ranlib generates an index to the contents of an archive, and stores it in the archive. The index lists each symbol defined by an archive member that is a relocatable object file.

readelf

readelf displays information about elf type binaries.

size

size lists the section sizes −−and the total size−− for each of the object files in its argument list. By default, one line of output is generated for each object file or each module in an archive.

strings

For each file given, strings prints the printable character sequences that are at least 4 characters long (or the number specified with an option to the program) and are followed by an unprintable character. By default, it only prints the strings from the initialized and loaded sections of object files. For other types of files, it prints the strings from the whole file.

strings is mainly useful for determining the contents of non−text files.

strip

strip discards all or specific symbols from object files. The list of object files may include archives. At least one object file must be given. strip modifies the files named in its argument, rather than writing modified copies under different names.

Library Files

libbfd.[a,so] and libopcodes.[a,so]

Descriptions

libbfd

libbfd is the Binary File Descriptor library.

libopcodes

libopcodes is a native library for dealing with opcodes and is used in the course of building utilities such as objdump. Opcodes are actually "readable text" versions of instructions for the processor.

Binutils Installation Dependencies

Last checked against version 2.11.2.

Autoconf: autoconf, autoheader Automake: aclocal, automake Bash: sh
Binutils: ar, as, ld, nm, ranlib, strip Diffutils: cmp
Fileutils: chmod, cp, ln, ls, mkdir, mv, rm, rmdir, touch Flex: flex
Gcc: cc, cc1, collect2, cpp0, gcc Glibc: ldconfig Grep: egrep, fgrep, grep M4: m4 Make: make
Gawk: gawk Sed: sed Sh−utils: basename, echo, expr, hostname, sleep, true, uname
Texinfo: install−info, makeinfo Textutils: cat, sort, tr, uniq

Bison

Official Download Location

Bison (1.35): ftp://ftp.gnu.org/gnu/bison/

Contents of Bison

Last checked against version 1.35.

Program Files

bison and yacc

Descriptions

bison

bison is a parser generator, a replacement for yacc. yacc stands for Yet Another Compiler Compiler. What is bison then? It is a program that generates a program that analyzes the structure of a text file. Instead of writing the actual program a user specifies how things should be connected and with those rules a program is constructed that analyzes the text file. There are a lot of examples where structure is needed and one of them is the calculator.

Given the string :

```
1 + 2 * 3
```

A human can easily come to the result 7. Why? Because of the structure. Our brain knows how to interpret the string. The computer doesn't know that and bison is a tool to help it understand by presenting the string in the following way to the compiler:

```
  +         / \         *   1         / \
2   3
```

Starting at the bottom of a tree and coming across the numbers 2 and 3 which are joined by the multiplication symbol, the computer multiplies 2 and 3. The result of that multiplication is remembered and the next thing that the computer sees is the result of 2*3 and the number 1 which are joined by the add symbol. Adding 1 to the previous result makes 7. In calculating, the most complex calculations can be broken down in this tree format and the computer just starts at the bottom and works its way up to the top and comes with the correct answer. Of course, bison isn't only used for calculators alone.

yacc

We create a bash script called yacc which calls bison using the −y option. This is for compatibility purposes for programs which use yacc instead of bison.

Bison Installation Dependencies

Last checked against version 1.31.

Bash: sh Binutils: ar, as, ld, ranlib Diffutils: cmp
Fileutils: chmod, cp, install, ln, ls, mkdir, mv, rm, rmdir Gcc: cc, cc1, collect2, cpp0, gcc
Grep: egrep, fgrep, grep Make: make Sed: sed
Sh−utils: basename, dirname, echo, expr, hostname, sleep, uname Texinfo: install−info
Textutils: cat, head, tr, uniq

Bzip2

Official Download Location

Bzip2 (1.0.2): ftp://sourceware.cygnus.com/pub/bzip2/

Contents of Bzip2

Last checked against version 1.0.2

Program Files

bunzip2 (link to bzip2), bzcat (link to bzip2), bzcmp, bzdiff, bzegrep, bzfgrep, bzgrep, bzip2, bzip2recover, bzless and bzmore

Descriptions

bunzip2

bunzip2 decompresses files that are compressed with bzip2.

bzcat

bzcat (or bzip2 −dc) decompresses all specified files to the standard output.

bzcmp, bzdiff

bzcmp and bzdiff are used to invoke the cmp or the diff program on bzip2 compressed files.

bzegrep, bzfgrep, bzgrep

bzegrep, bzfgrep, and bzgrep invoke either egrep, fgrep, or grep (respectively) on bzip2−compressed files.

bzip2

bzip2 compresses files using the Burrows−Wheeler block sorting text compression algorithm and Huffman coding. Compression is generally considerably better than that achieved by more conventional LZ77/LZ78−based compressors and approaches the performance of the PPM family of statistical compressors.

bzip2recover

bzip2recover recovers data from damaged bzip2 files.

bzless

bzless is a filter which allows examination of compressed or plain text files, one screenful at a time on a soft−copy terminal, like less.

bzmore

bzmore is a filter which allows examination of compressed or plain text files, one screenful at a time on a soft−copy terminal, like more.

Library Files

libbz2.a, libbz2.so (link to libbz2.so.1.0), libbz2.so.1.0 (link to libbz2.so.1.0.2) and libbz2.so.1.0.2

libbz2

libbz2 is the library for implementing lossless, block−sorting data compression, using the Burrows−Wheeler algorithm.

Bzip2 Installation Dependencies

Last checked against version 1.0.1.

Bash: sh Binutils: ar, as, ld, ranlib Fileutils: cp, ln, rm Gcc: cc1, collect2, cpp0, gcc Make: make

Diffutils

Official Download Location

Diffutils (2.8.1): ftp://ftp.gnu.org/gnu/diffutils/

Contents of Diffutils

Last checked against version 2.8.1.

Program Files

cmp, diff, diff3 and sdiff

Descriptions

cmp and diff

cmp and diff both compare two files and report their differences. Both programs have extra options which compare files in different situations.

diff3

The difference between diff and diff3 is that diff compares 2 files, diff3 compares 3 files.

sdiff

sdiff merges two files and interactively outputs the results.

Diffutils Installation Dependencies

Last checked against version 2.7.

Bash: sh Binutils: ld, as Diffutils: cmp Fileutils: chmod, cp, install, mv, rm
Gcc: cc1, collect2, cpp0, gcc Grep: egrep, grep Make: make Sed: sed Sh–utils: date, hostname
Textutils: cat, tr

E2fsprogs

Official Download Location

E2fsprogs (1.27): ftp://download.sourceforge.net/pub/sourceforge/e2fsprogs/
http://download.sourceforge.net/e2fsprogs/

Contents of E2fsprogs

Last checked against version 1.27.

Program Files

badblocks, chattr, compile_et, debugfs, dumpe2fs, e2fsck, e2image, e2label, fsck, fsck.ext2, fsck.ext3, lsattr, mk_cmds, mke2fs, mkfs.ext2, mkfs.ext3, mklost+found, resize2fs, tune2fs and uuidgen

Descriptions

badblocks

badblocks is used to search for bad blocks on a device (usually a disk partition).

chattr

chattr changes the file attributes on a Linux second extended file system.

compile_et

compile_et is used to convert a table, listing error−code names and associated messages, into a C source file that is suitable for use with the com_err library.

debugfs

The debugfs program is a file system debugger. It can be used to examine and change the state of an ext2 file system.

dumpe2fs

dumpe2fs prints the super block and blocks group information for the filesystem present on a specified device.

e2fsck and fsck.ext2

e2fsck and fsck.ext2 are used to check, and optionally repair, Linux second extended filesystems.

e2image

e2image is used to save critical ext2 filesystem data to a file.

e2label

e2label will display or change the filesystem label on the ext2 filesystem located on the specified device.

fsck

fsck is used to check, and optionally repair, a Linux file system.

fsck.ext3

fsck.ext3 is used to check, and optionally repair, a Linux ext3 filesystems.

lsattr

lsattr lists the file attributes on a second extended file system.

mk_cmds

The mk_cmds utility takes a command table file as input and produces a C source file as output, which is intended to be used with the subsystem library, libss.

mke2fs and mkfs.ext2

mke2fs is used to create a Linux second extended file system on a device (usually a disk partition). mkfs.ext2 does the same as mke2fs.

mkfs.ext3

mkfs.ext3 is used to create an ext3 filesystem.

mklost+found

mklost+found is used to create a lost+found directory in the current working directory on a Linux second extended file system. mklost+found pre−allocates disk blocks to the directory to make it usable by e2fsck.

resize2fs

resize2fs is used to resize ext2 file systems.

tune2fs

tune2fs adjusts tunable filesystem parameters on a Linux second extended filesystem.

uuidgen

The uuidgen program creates a new universally unique identifier (UUID) using the libuuid library. The 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.

Library Files

libcom_err.[a,so], libe2p.[a,so], libext2fs.[a,so], libss.[a,so], libuuid.[a,so]

Descriptions

libcom_err

The common error display routine.

libe2p

libe2p is used by dumpe2fs, chattr, and lsattr.

libext2fs

The ext2fs library is designed to allow user−level programs to manipulate an ext2 filesystem.

libss

libss is used by debugfs.

libuuid

The libuuid library is used to generate unique identifiers for objects that may be accessible beyond the local system.

E2fsprogs Installation Dependencies

Last checked against version 1.25.

Bash: sh Binutils: ar, as, ld, ranlib, strip Diffutils: cmp
Fileutils: chmod, cp, install, ln, mkdir, mv, rm, sync Gcc: cc, cc1, collect2, cpp0 Glibc: ldconfig
Grep: egrep, grep Gzip: gzip Make: make Gawk: awk Sed: sed
Sh−utils: basename, echo, expr, hostname, uname Texinfo: makeinfo Textutils: cat, tr

Ed

Official Download Location

Ed (0.2): ftp://ftp.gnu.org/gnu/ed/ Ed Patch (0.2): ftp://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/ http://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/

Contents of Ed

Last checked against version 0.2.

Program Files

ed and red (link to ed)

Description

ed

ed is a line−oriented text editor. It is used to create, display, modify and otherwise manipulate text files.

red

red is a restricted ed: it can only edit files in the current directory and cannot execute shell commands.

Ed Installation Dependencies

Last checked against version 0.2.

Bash: sh Binutils: ar, as, ld, ranlib Diffutils: cmp Fileutils: chmod, cp, install, ln, mv, rm, touch
Gcc: cc1, collect2, cpp0, gcc Grep: egrep, grep Make: make Sed: sed Sh−utils: hostname
Textutils: cat, tr

File

Official Download Location

File (3.39): ftp://ftp.gw.com/mirrors/pub/unix/file/

Contents of File

Last checked against version 3.39.

Program Files

file

Descriptions

file

file tests each specified file in an attempt to classify it. There are three sets of tests, performed in this order: filesystem tests, magic number tests and language tests. The first test that succeeds causes the file type to be printed.

File Installation Dependencies

Last checked against version 3.37.

Autoconf: autoconf, autoheader Automake: aclocal, automake Bash: sh Binutils: as, ld Diffutils: cmp
Fileutils: chmod, install, ln, ls, mv, rm, touch Gcc: cc1, collect2, cpp0, gcc Grep: egrep, grep
M4: m4 Make: make Gawk: gawk Sed: sed Sh−utils: echo, expr, hostname, sleep Texinfo: makeinfo
Textutils: cat, tr

Fileutils

Official Download Location

Fileutils (4.1): ftp://ftp.gnu.org/gnu/fileutils/ Fileutils Patch (4.1): ftp://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/ http://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/

Contents of Fileutils

Last checked against version 4.1.

Program Files

chgrp, chmod, chown, cp, dd, df, dir, dircolors, du, install, ln, ls, mkdir, mkfifo, mknod, mv, rm, rmdir, shred, sync, touch and vdir

Descriptions

chgrp

chgrp changes the group ownership of each given file to the named group, which can be either a group name or a numeric group ID.

chmod

chmod changes the permissions of each given file according to mode, which can be either a symbolic representation of changes to make or an octal number representing the bit pattern for the new permissions.

chown

chown changes the user and/or group ownership of each given file.

cp

cp copies files from one place to another.

dd

dd copies a file (from the standard input to the standard output, by default) with a user−selectable blocksize, while optionally performing conversions on it.

df

df displays the amount of disk space available on the filesystem containing each file name argument. If no file name is given, the space available on all currently mounted filesystems is shown.

dir, ls and vdir

dir and vdir are versions of ls with different default output formats. These programs list each given file or directory name. Directory contents are sorted alphabetically. For ls, files are, by default, listed in columns sorted vertically if the standard output is a terminal; otherwise they are listed one per line. For dir, files are, by default, listed in columns sorted vertically. For vdir, files are, by default, listed in long format.

dircolors

dircolors outputs commands to set the LS_COLOR environment variable. The LS_COLOR variable is use to change the default color scheme used by ls and related utilities.

du

du displays the amount of disk space used by each file or directory listed on the command–line and by each of their subdirectories.

install

install copies files and sets their permission modes and, if possible, their owner and group.

ln

ln makes hard or soft (symbolic) links between files.

mkdir

mkdir creates directories with a given name.

mkfifo

mkfifo creates a FIFO with each given name.

mknod

mknod creates a FIFO, character special file or block special file with the given file name.

mv

mv moves files from one directory to another or renames files, depending on the arguments given to mv.

rm

rm removes files or directories.

rmdir

rmdir removes directories, if they are empty.

shred

shred deletes a file securely, overwriting it first so that its contents can't be recovered.

sync

sync forces changed blocks to disk and updates the super block.

touch

touch changes the access and modification times of each given file to the current time. Files that do not exist are created empty.

Fileutils Installation Dependencies

Last checked against version 4.1.

Bash: sh Binutils: ar, as, ld, ranlib Diffutils: cmp
Fileutils: chmod, cp, install, ln, ls, mkdir, mv, rm, rmdir Gettext: msgfmt, xgettext
Gcc: cc, cc1, collect2, cpp0, gcc Grep: egrep, fgrep, grep Make: make Perl: perl Sed: sed
Sh–utils: basename, echo, expr, hostname, sleep, uname Texinfo: install–info Textutils: cat, tr

Findutils

Official Download Location

Findutils (4.1): ftp://ftp.gnu.org/gnu/findutils/ Findutils Patch (4.1): ftp://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/ http://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/

Contents of Findutils

Last checked against version 4.1.

Program Files

bigram, code, find, frcode, locate, updatedb and xargs

Descriptions

bigram

bigram is used together with code to produce older−style locate databases. To learn more about these last three programs, read the locatedb.5 manual page.

code

code is the ancestor of frcode. It was used in older−style locate databases.

find

The find program searches for files in a directory hierarchy which match a certain criteria. If no criteria is given, it lists all files in the current directory and its subdirectories.

frcode

frcode is called by updatedb to compress the list of file names using front−compression, which reduces the database size by a factor of 4 to 5.

locate

locate scans a database which contains all files and directories on a filesystem. This program lists the files and directories in this database matching a certain criteria. If a user is looking for a file this program will scan the database and tell him exactly where the files he requested are located. This only makes sense if the locate database is fairly up−to−date, else it will provide out−of−date information.

updatedb

The updatedb program updates the locate database. It scans the entire file system (including other file systems that are currently mounted unless it is told not to do so) and puts every directory and file it finds into the database that's used by the locate program, which retrieves this information. It's good practice to update this database once a day to have it up−to−date whenever it is needed.

xargs

The xargs command applies a command to a list of files. If there is a need to perform the same command on multiple files, a list can be created that names all those files (one per line) and xargs can perform that command on those files.

Findutils Installation Dependencies

Last checked against version 4.1.

Bash: sh Binutils: ar, as, ld, ranlib Diffutils: cmp Fileutils: chmod, cp, install, mv, rm
Grep: egrep, grep Gcc: cc1, collect2, cpp0, gcc Make: make Patch: patch Sed: sed
Sh−utils: basename, date, echo, hostname Textutils: cat, tr

Flex

Official Download Location

Flex (2.5.4a): ftp://ftp.gnu.org/non−gnu/flex/

Contents of Flex

Last checked against version 2.5.4a.

Program Files

flex, flex++ (link to flex) and lex

Descriptions

flex

flex is a tool for generating programs which recognize patterns in text. Pattern recognition is very useful in many applications. A user sets up rules about what to look for and flex will make a program that looks for those patterns. The reason people use flex is that it is much easier to set up rules for what to look for than to write the actual program which finds the text.

flex++

flex++ invokes a version of flex which is used exclusively for C++ scanners.

lex

We create a bash script called lex which calls flex using the −l option. This is for compatibility purposes for programs which use lex instead of flex.

Library Files

libfl.a

Descriptions

libfl

libfl is the flex library.

Flex Installation Dependencies

Last checked against version 2.5.4a.

Bash: sh Binutils: ar, as, ld, ranlib Bison: bison Diffutils: cmp
Fileutils: chmod, cp, install, ln, mv, rm, touch Gcc: cc1, collect2, cpp0, gcc Grep: egrep, grep
Make: make Sed: sed Sh−utils: echo, hostname Textutils: cat, tr

Gawk

Official Download Location

Gawk (3.1.1): ftp://ftp.gnu.org/pub/gnu/gawk/ Gawk Patch (3.1.1−2):
ftp://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/ http://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/

Contents of Gawk

Last checked against version 3.1.1.

Program Files

awk, gawk, gawk−3.1.1, grcat, igawk, pgawk, pgawk−3.1.1, pwcat

Descriptions

awk

awk is a symbolic link to gawk.

gawk, gawk−3.1.1

gawk is the GNU implementation of awk, a pattern scanning and processing language.

grcat

grcat concatenates the group database, /etc/group.

igawk

igawk is a shell script which gives gawk the ability to include files.

pgawk, pgawk−3.1.1

pgawk is the profiling version of gawk.

pwcat

pwcat concatenates the password database, /etc/passwd.

Gawk Installation Dependencies

Last checked against version 3.1.0.

(No dependencies checked yet.)

GCC

Official Download Location

GCC (3.2): ftp://ftp.gnu.org/pub/gnu/gcc/ GCC Patch (3.2): ftp://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/ http://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/

Contents of GCC

Last checked against version 3.1.

Program Files

c++, c++filt, cc (link to gcc), cc1, cc1plus, collect2, cpp, cpp0, g++, gcc, gccbug, gcov and tradcpp0

Descriptions

cc, cc1, cc1plus, gcc

These are the C compiler. A compiler translates source code in text format to a format that a computer understands. After a source code file is compiled into an object file, a linker will create an executable file from one or more of these compiler generated object files.

c++, cc1plus, g++

These are the C++ compiler, the equivalent of cc and gcc etc.

c++filt

The C++ language provides function overloading, which means that it is possible to write many functions with the same name (providing each takes parameters of different types). All C++ function names are encoded into a low−level assembly label (this process is known as mangling). The c++filt program does the inverse mapping: it decodes (demangles) low−level names into user−level names so that the linker can keep these overloaded functions from clashing.

collect2

collect2 assists with the compilation of constructors.

cpp, cpp0

cpp pre−processes a source file, such as including the contents of header files into the source file. Simply add a line, such as #include <filename>, to your source file. The preprocessor will insert the contents of the included file into the source file.

gccbug

gccbug is a shell script which is used to simplify the creation of bug reports.

gcov

gcov analyzes programs to help create more efficient, faster running code through optimization.

tradcpp0

No description is currently available.

Library Files

libgcc.a, libgcc_eh.a, libgcc_s.so, libiberty.a, libstdc++.[a,so], libsupc++.a

Descriptions

libgcc, libgcc_eh, libgcc_s

Run−time support files for gcc.

libiberty

libiberty is a collection of subroutines used by various GNU programs including getopt, obstack, strerror, strtol and strtoul.

libstdc++

libstdc++ is the C++ library. It is used by C++ programs and contains functions that are frequently used in C++ programs. This way the programmer doesn't have to write certain functions (such as writing a string of text to the screen) from scratch every time he creates a program.

libsupc++

libsupc++ provides support for the c++ programming language. Among other things, libsupc++ contains routines for exception handling.

GCC Installation Dependencies

Last checked against version 2.95.3.

Bash: sh Binutils: ar, as, ld, nm, ranlib Diffutils: cmp
Fileutils: chmod, cp, ln, ls, mkdir, mv, rm, touch Find: find Gcc: cc, cc1, collect2, cpp0, gcc
Grep: egrep, grep Make: make Patch: patch Sed: sed
Sh−utils: basename, dirname, echo, expr, hostname, sleep, true, uname Tar: tar
Texinfo: install−info, makeinfo Textutils: cat, tail, tr

Gettext

Official Download Location

Gettext (0.11.5): ftp://ftp.gnu.org/gnu/gettext/

Contents of Gettext

Last checked against version 0.11.2.

Program Files

config.charset, config.rpath, gettext, gettextize, hostname, msgattrib, msgcat, msgcmp, msgcomm, msgconv, msgen, msgexec, msgfilter, msgfmt, msggrep, msginit, msgmerge, msgunfmt, msguniq, ngettext, project−id, team−address, trigger, urlget, user−email and xgettext

Descriptions

config.charset

The config.charset script outputs a system−dependent table of character encoding aliases.

config.rpath

The config.rpath script outputs a system−dependent set of variables, describing how to set the run time search path of shared libraries in an executable.

gettext

The gettext package is used for internationalization (also known as i18n) and for localization (also known as l10n). Programs can be compiled with Native Language Support (NLS) which enable them to output messages in the user's native language rather than in the default English language.

gettextize

The gettextize program copies all standard gettext files into a directory. It's used to make a package with gettext translations.

hostname

The hostname program displays a network hostname in various forms.

msgattrib

The msgattrib program filters the messages of a translation catalog according to their attributes and manipulates the attributes.

msgcat

The msgcat program finds messages which are common in several raw translations.

msgcmp

The msgcmp program compares two raw translation files.

msgcomm

The msgcomm program searches messages which appear in several .po files. It's used to compare how things are translated.

msgconv

The msgconv program converts a translation catalog to a different character encoding.

msgen

The msgen program creates an English translation catalog.

msgexec

The msgexec program applies a command to all translations of a translation catalog.

msgfilter

The msgfilter program applies a filter to all translations of a translation catalog.

msgfmt

The msgfmt program compiles raw translation into machine code. It's used to create the final program/package translation file.

msggrep

The msggrep program extracts all messages of a translation catalog that match a given pattern or belong to some given source files.

msginit

The msginit program creates a new PO file, initializing the meta information with values from the user's environment.

msgmerge

The msgmerge program combines two raw translations into one file. It's used to update the raw translation with the source extract.

msgunfmt

The msgunfmt program decompiles translation files into raw translation text. It can only be used if the compiled versions are available.

msguniq

The msguniq program unifies duplicate translations in a translation catalog.

ngettext

The ngettext program displays native language translations of a textual message whose grammatical form depends on a number.

project−id

The project−id script prints a package's identification package version or package.

team−address

The team−address script prints the team's address to stdout and outputs additional instructions.

trigger

The trigger script tests whether the current package is a GNOME or KDE package.

urlget

The urlget program gets the contents of a URL.

user−email

The user−email script prints the user's email address, with confirmation from the user.

xgettext

The xgettext program extracts the message lines from the programmers' C files. It's used to make the first translation template.

Library Files

libgettextlib[a,so], libgettextsrc[a,so]

Descriptions

libgettextlib

No description is currently available.

libgettextsrc

No description is currently available.

Gettext Installation Dependencies

Last checked against version 0.10.40.

Autoconf: autoconf, autoheader Automake: aclocal, automake Bash: sh
Binutils: ar, as, ld, nm, ranlib, strip Bison: bison Diffutils: cmp
Fileutils: chmod, install, ln, ls, mkdir, mv, rm, rmdir Gcc: cc, cc1, collect2, cpp0, gcc
Grep: egrep, fgrep, grep M4: m4 Make: make Gawk: gawk Sed: sed
Sh−utils: basename, echo, expr, hostname, sleep, uname Texinfo: install−info, makeinfo
Textutils: cat, sort, tr, uniq

Glibc

Official Download Location

Glibc (2.2.5): ftp://ftp.gnu.org/gnu/glibc/ Glibc−linuxthreads (2.2.5): ftp://ftp.gnu.org/gnu/glibc/
Glibc Patch (2.2.5−2): ftp://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/
http://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/

Contents of Glibc

Last checked against version 2.2.5.

Program Files

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

Descriptions

catchsegv

catchsegv can be used to create a stack trace when a program terminates with a segmentation fault.

gencat

gencat generates message catalogues.

getconf

getconf displays the system configuration values for filesystem specific variables.

getent

getent gets entries from an administrative database.

glibcbug

glibcbug creates a bug report about glibc and and mails it to the bug email address.

iconv

iconv performs character set conversion.

iconvconfig

iconvconfig creates fastloading iconv module configuration file.

ldconfig

ldconfig configures the dynamic linker run time bindings.

ldd

ldd prints the shared libraries required by each program or shared library specified on the command line.

lddlibc4

lddlibc4 assists ldd with object files.

locale

locale is a Perl program which tells the compiler to enable (or disable) the use of POSIX locales for built–in operations.

localedef

localedef compiles locale specifications.

mtrace

mtrace prints the multicast path from a source to a receiver (an IP trace query).

nscd

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

nscd_nischeck

nscd_nischeck checks whether or not secure mode is necessary for NIS+ lookup.

pcprofiledump

pcprofiledump dumps information generated by PC profiling.

pt_chown

pt_chown sets the owner, group and access permission of the slave pseudo terminal corresponding to the master pseudo terminal passed on file descriptor `3'. This is the helper program for the `grantpt' function. It is not intended to be run directly from the command line.

rpcgen

rpcgen generates C code to implement the RPC protocol.

rpcinfo

rpcinfo makes an RPC call to an RPC server.

sln

sln symbolically links dest to source. It is statically linked, needing no dynamic linking at all. Thus sln is useful to make symbolic links to dynamic libraries if the dynamic linking system for some reason is nonfunctional.

sprof

sprof reads and displays shared object profiling data.

tzselect

tzselect asks the user for information about the current location and outputs the resulting time zone description to standard output.

xtrace

xtrace traces execution of program by printing the currently executed function.

zdump

zdump is the time zone dumper.

zic

zic is the time zone compiler.

Library Files

ld.so, libBrokenLocale.[a,so], libBrokenLocale_p.a, libSegFault.so, libanl.[a,so], libanl_p.a, libbsd−compat.a, libc.[a,so], libc_nonshared.a, libc_p.a, libcrypt.[a,so], libcrypt_p.a, libdl.[a,so], libdl_p.a, libg.a, libieee.a, libm.[a,so], libm_p.a, libmcheck.a, libmemusage.so, libnsl.a, libnsl_p.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_p.a, libresolv.[a,so], libresolv_p.a, librpcsvc.a, librpcsvc_p.a, librt.[a,so], librt_p.a, libthread_db.so, libutil.[a,so] and libutil_p.a

Descriptions

ld.so

ld.so is the helper program for shared library executables.

libBrokenLocale, libBrokenLocale_p

Used by software, such as Mozilla, to solve broken locales.

libSegFault

libSegFault is a segmentation fault signal handler. It tries to catch segfaults.

libanl, libanl_p

libanl is an asynchronous name lookup library.

libbsd−compat

libbsd−compat provides the portability needed in order to run certain programs in Linux.

libc, libc_nonshared, libc_p

These files constitute the main C library. The C library is a collection of commonly used functions in programs. This way a programmer doesn't need to create his own functions for every single task. The most common things like writing a string to the screen are already present and at the disposal of the programmer.

The C library (actually almost every library) comes in two flavors: a dynamic and a static one. In short, when a program uses a static C library, the code from the C library is copied into the executable file. When a program uses a dynamic library, the executable will not contain the code from the C library, but instead a routine that loads the functions from the library at the time the program is run. This means a significant decrease in the file size of a program. The documentation that comes with the C library describes this in more detail, as it is too complicated to explain here in one or two lines.

libcrypt, libcrypt_p

libcrypt is the cryptography library.

libdl, libdl_p

libdl is the dynamic linking interface library.

libg

libg is a runtime library for g++.

libieee

libieee is the IEEE floating point library.

libm, libm_p

libm is the mathematical library.

libmcheck

libmcheck contains code run at boot.

libmemusage

libmemusage is used by memusage to help collect information about the memory usage of a program.

libnsl, libnsl_p

libnsl is the network services library.

libnss_compat, libnss_dns, libnss_files, libnss_hesiod, libnss_nis, libnss_nisplus

The basic idea is to put the implementation of the different services offered to access the databases in separate modules. This has some advantages:

- contributors can add new services without adding them to GNU C library,
- the modules can be updated separately,
- the C library image is smaller.

libpcprofile

Code used by the kernel to track CPU time spent in functions, source code lines, and instructions.

libpthread, libpthread_p

The POSIX threads library.

libresolv, libresolv_p

Functions in this library provide for creating, sending, and interpreting packets to the Internet domain name servers.

librpcsvc, librpcsvc_p

Functions in this library provide miscellaneous RPC services.

librt, librt_p

Functions in this library provide most of the interfaces specified by the POSIX.1b Realtime Extension.

libthread_db

Functions is this library are useful for building debuggers for multi−threaded programs.

libutil, libutil_p

Contains code for "standard" functions used in many different Unix utilities.

Glibc Installation Dependencies

Last checked against version 2.2.5.

Bash: sh Binutils: ar, as, ld, ranlib, readelf Diffutils: cmp
Fileutils: chmod, cp, install, ln, mknod, mv, mkdir, rm, touch Gcc: cc, cc1, collect2, cpp, gcc
Grep: egrep, grep Gzip: gzip Make: make Gawk: gawk Sed: sed
Sh−utils: date, expr, hostname, pwd, uname Texinfo: install−info, makeinfo
Textutils: cat, cut, sort, tr

Grep

Official Download Location

Grep (2.5): ftp://ftp.gnu.org/gnu/grep/

Contents of Grep

Last checked against version 2.5.

Program Files

egrep (link to grep), fgrep (link to grep) and grep

Descriptions

egrep

egrep prints lines from files matching an extended regular expression pattern.

fgrep

fgrep prints lines from files matching a list of fixed strings, separated by newlines, any of which is to be matched.

grep

grep prints lines from files matching a basic regular expression pattern.

Grep Installation Dependencies

Last checked against version 2.4.2.

Autoconf: autoconf, autoheader Automake: aclocal, automake Bash: sh Binutils: as, ld Diffutils: cmp
Fileutils: chmod, install, ls, mkdir, mv, rm Gettext: msgfmt, xgettext
Gcc: cc, cc1, collect2, cpp0, gcc Glibc: getconf Grep: egrep, fgrep, grep M4: m4 Make: make
Gawk: gawk Sed: sed Sh−utils: basename, echo, expr, hostname, sleep, uname
Texinfo: install−info, makeinfo Textutils: cat, tr

Groff

Official Download Location

Groff (1.18): ftp://ftp.gnu.org/gnu/groff/

Contents of Groff

Last checked against version 1.17.2.

Program Files

addftinfo, afmtodit, eqn, geqn (link to eqn), grn, grodvi, groff, grog, grolbp, grolj4, grops, grotty, gtbl (link to tbl), hpftodit, indxbib, lkbib, lookbib, mmroff, neqn, nroff, pfbtops, pic, post−grohtml, pre−grohtml, refer, soelim, tbl, tfmtodit, troff and zsoelim (link to soelim)

Descriptions

addftinfo

addftinfo reads a troff font file and adds some additional font−metric information that is used by the groff system.

afmtodit

afmtodit creates a font file for use with groff and grops.

eqn

eqn compiles descriptions of equations embedded within troff input files into commands that are understood by troff.

geqn

geqn is the GNU implementation of eqn.

grn

grn is a groff preprocessor for gremlin files.

grodvi

grodvi is a driver for groff that produces TeX dvi format.

groff

groff is a front−end to the groff document formatting system. Normally it runs the troff program and a post−processor appropriate for the selected device.

grog

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 prints the groff command including those options on the standard output.

grolbp

grolbp is a groff driver for Canon CAPSL printers (LBP−4 and LBP−8 series laser printers).

grolj4

grolj4 is a driver for groff that produces output in PCL5 format suitable for an HP Laserjet 4 printer.

grops

grops translates the output of GNU troff to Postscript.

grotty

grotty translates the output of GNU troff into a form suitable for typewriter−like devices.

gtbl

gtbl is the GNU implementation of tbl.

hpftodit

hpftodit creates a font file for use with groff −Tlj4 from an HP tagged font metric file.

indxbib

indxbib makes an inverted index for the bibliographic databases a specified file for use with refer, lookbib, and lkbib.

lkbib

lkbib searches bibliographic databases for references that contain specified keys and prints any references found on the standard output.

lookbib

lookbib prints a prompt on the standard error (unless the standard input is not a terminal), reads from the standard input a line containing a set of keywords, 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

mmroff is a simple preprocessor for groff.

neqn

The neqn script formats equations for ascii output.

nroff

The nroff script emulates the nroff command using groff.

pfbtops

pfbtops translates a Postscript font in .pfb format to ASCII.

pic

pic compiles descriptions of pictures embedded within troff or TeX input files into commands that are understood by TeX or troff.

pre−grohtml and post−grohtml

pre− and post−grohtml translate the output of GNU troff to html.

refer

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 about how citations are to be processed.

soelim

soelim reads files and replaces lines of the form *.so file* by the contents of *file*.

tbl

tbl compiles descriptions of tables embedded within troff input files into commands that are understood by troff.

tfmtodit

tfmtodit creates a font file for use with **`groff −Tdvi`**.

troff

troff is highly compatible with Unix troff. Usually it should be invoked using the groff command, which will also run preprocessors and post−processors in the appropriate order and with the appropriate options.

zsoelim

zsoelim is the GNU implementation of soelim.

Groff Installation Dependencies

Last checked against version 1.17.2.

Bash: sh Binutils: ar, as, ld, ranlib Bison: bison Diffutils: cmp
Fileutils: chmod, cp, install, ln, ls, mkdir, mv, rm, touch
Gcc: cc1, cc1plus, collect2, cpp0, g++, gcc Grep: egrep, grep Make: make Gawk: awk Sed: sed
Sh−utils: basename, date, echo, expr, hostname, uname Textutils: cat, tr

Gzip

Official Download Location

Gzip (1.2.4a): ftp://ftp.gnu.org/gnu/gzip/

Contents of Gzip

Last checked against version 1.2.4a.

Program Files

gunzip (link to gzip), gzexe, gzip, uncompress (link to gunzip), zcat (link to gzip), zcmp, zdiff, zforce, zgrep, zmore and znew

Description

gunzip, uncompress

gunzip and uncompress decompress files which are compressed with gzip.

gzexe

gzexe allows you to compress executables in place and have them automatically uncompress and execute when they are run (at a penalty in performance).

gzip

gzip reduces the size of the named files using Lempel–Ziv coding (LZ77).

zcat

zcat uncompresses, and writes to standard output, either a list of files on the command line or a file being read from standard input.

zcmp

zcmp invokes the cmp program on compressed files.

zdiff

zdiff invokes the diff program on compressed files.

zforce

zforce forces a .gz extension on all gzip files so that gzip will not compress them twice. This can be useful for files with names truncated after a file transfer.

zgrep

zgrep invokes the grep program on compressed files.

zmore

zmore is a filter which allows examination of compressed or plain text files, one screen at a time on a soft−copy terminal (similar to the more program).

znew

znew re−compresses files from .Z (compress) format to .gz (gzip) format.

Gzip Installation Dependencies

Last checked against version 1.2.4a.

Bash: sh Binutils: as, ld, nm Fileutils: chmod, cp, install, ln, mv, rm
Gcc: cc1, collect2, cpp, cpp0, gcc Grep: egrep, grep Make: make Sed: sed Sh−utils: hostname
Textutils: cat, tr

Kbd

Official Download Location

Kbd (1.06): ftp://ftp.win.tue.nl/pub/linux−local/utils/kbd/ Kbd Patch (1.06−3): ftp://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/ http://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/

Contents of Kbd

Last checked against version 1.06.

Program Files

chvt, deallocvt, dumpkeys, fgconsole, getkeycodes, getunimap, kbd_mode, kbdrate, loadkeys, loadunimap, mapscrn, openvt, psfaddtable (link to psfxtable), psfgettable (link to psfxtable), psfstriptable (link to psfxtable), psfxtable, resizecons, setfont, setkeycodes, setleds, setlogcons, setmetamode, setvesablank, showfont, showkey, unicode_start, and unicode_stop

Descriptions

chvt

chvt changes foreground virtual terminal.

deallocvt

deallocvt deallocates unused virtual terminals.

dumpkeys

dumpkeys dumps keyboard translation tables.

fgconsole

fgconsole prints the number of the active virtual terminal.

getkeycodes

getkeycodes prints the kernel scancode−to−keycode mapping table.

getunimap

getunimap prints the currently used unimap.

kbd_mode

kbd_mode reports or sets the keyboard mode.

kbdrate

kbdrate sets the keyboard repeat and delay rates.

loadkeys

loadkeys loads keyboard translation tables.

loadunimap

loadunimap loads the kernel unicode–to–font mapping table.

mapscrn

mapscrn loads a user defined output character mapping table into the console driver. Note that it is obsolete and that its features are built into setfont.

openvt

openvt starts a program on a new virtual terminal (VT).

psfaddtable, psfgettable, psfstriptable, psfxtable

These are a set of tools for handling Unicode character tables for console fonts.

resizecons

resizecons changes the kernel idea of the console size.

setfont

This lets you change the EGA/VGA fonts in console.

setkeycodes

setkeycodes loads kernel scancode–to–keycode mapping table entries.

setleds

setleds sets the keyboard LEDs. Many people find it useful to have numlock enabled by default and, by using this program, you can achieve this.

setlogcons

setlogcons sends kernel messages to the console.

setmetamode

setmetamode defines the keyboard meta key handling.

setvesablank

This lets you fiddle with the built−in hardware screensaver (not toasters, only a blank screen).

showfont

showfont displays data about a font. The information shown includes font information, font properties, character metrics and character bitmaps.

showkey

showkey examines the scancodes and keycodes sent by the keyboard.

unicode_start

unicode_start puts the console in Unicode mode.

unicode_stop

unicode_stop reverts keyboard and console from unicode mode.

Kbd Installation Dependencies

Last checked against version 1.06.

Bash: sh Binutils: as, ld, strip Bison: bison Diffutils: cmp Fileutils: cp, install, ln, mv, rm Flex: flex Gettext: msgfmt, xgettext Gcc: cc1, collect2, cpp0, gcc Grep: grep Gzip: gunzip, gzip Make: make Patch: patch Sed: sed Sh−utils: uname

Less

Official Download Location

Less (374): ftp://ftp.gnu.org/gnu/less/

Contents of Less

Last checked against version 374.

Program Files

less, lessecho and lesskey

Descriptions

less

The less program is a file pager (or text viewer). It displays the contents of a file and has the ability to scroll. Less is an improvement on the common pager called "more". Less has the ability to scroll backwards through files as well and it doesn't need to read the entire file when it starts, which makes it faster when reading large files.

lessecho

lessecho is needed to expand metacharacters, such as * and ?, in filenames on Unix systems.

lesskey

lesskey is used to specify key bindings for less.

Less Installation Dependencies

Last checked against version 358.

Bash: sh Binutils: as, ld Diffutils: cmp Fileutils: chmod, install, mv, rm, touch Grep: egrep, grep
Gcc: cc1, collect2, cpp0, gcc Make: make Sed: sed Sh−utils: expr, hostname, uname Textutils: cat, tr

LFS−Bootscripts

Official Download Location

LFS−Bootscripts (1.10): ftp://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/
http://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/

Contents of LFS−bootscripts

Last checked against version 1.10.

Scripts

checkfs, cleanfs, functions, halt, ifdown, ifup, loadkeys, localnet, mountfs, mountproc, network, rc, reboot, sendsignals, setclock, swap, sysklogd and template

Descriptions

checkfs

The checkfs script checks the file systems just before they are mounted (with the exception of journal and network based file systems).

cleanfs

The cleanfs script removes files that shouldn't be preserved between reboots, such as /var/run/*, /var/lock/*, it re−creates /var/run/utmp and removes the possible present /etc/nologin, /fastboot and /forcefsck files.

functions

The functions script contains shared functions among different scripts such as error checking, status checking, etc.

halt

The halt script halts the system.

ifdown, ifup

The ifdown and ifup scripts assist the network script with network devices.

loadkeys

The loadkeys script loads the proper keymap table that matches your keyboard layout.

localnet

The localnet script sets up the system's hostname and local loopback device.

mountfs

The mountfs script mounts all file systems that aren't marked noauto or aren't network based.

mountproc

The mountproc script is used to mount the proc filesystem.

network

The network script sets up network interfaces (such as network cards) and sets up the default gateway where applicable.

rc

The rc script is the master runlevel control script which is responsible for running all the other scripts one−by−one in a specific sequence.

reboot

The reboot scripts reboots the system.

sendsignals

The sendsignals script makes sure every process is terminated before the system reboots or halts.

setclock

The setclock scripts resets the kernel clock to localtime in case the hardware clock isn't set to GMT time.

swap

The swap scripts enables and disables swap files and partitions.

sysklogd

The sysklogd script start and stops the system and kernel log daemons.

template

The template script is a template you can use to create your own bootscripts for your other daemons.

LFS−Bootscripts Installation Dependencies

Last checked against version 1.10.

Fileutils: chown, cp

Libtool

Official Download Location

Libtool (1.4.2): ftp://ftp.gnu.org/gnu/libtool/

Contents of Libtool

Last checked against version 1.4.2.

Program Files

libtool and libtoolize

Descriptions

libtool

libtool provides generalized library−building support services.

libtoolize

libtoolize provides a standard way to add libtool support to a package.

Library Files

libltdl.a, libltdl.so (link to libltdl.so.3.1.0), libltdl.so.3 (link to libltdl.so.3.1.0) and libltdl.so.3.1.0

Descriptions

libltdl, libltdl.so.3, libltdl.so.3.1.0

A small library that aims at hiding, from programmers, the various difficulties of dlopening libraries.

Libtool Installation Dependencies

Last checked against version 1.4.2.

Bash: sh Binutils: ar, as, ld, nm, ranlib, strip Diffutils: cmp
Fileutils: chmod, cp, install, ln, ls, mkdir, mv, rm, rmdir Gcc: cc, cc1, collect2, cpp0
Glibc: ldconfig Grep: egrep, fgrep, grep Make: make Sed: sed
Sh−utils: echo, expr, hostname, sleep, uname Texinfo: install−info Textutils: cat, sort, tr, uniq

Lilo

Official Download Location

Lilo (22.2): ftp://ibiblio.org/pub/Linux/system/boot/lilo/ http://ibiblio.org/pub/Linux/system/boot/lilo/

Contents of Lilo

Last checked against version 22.2.

Program Files

lilo, mkrescue and keytab−lilo.pl

Descriptions

lilo

lilo installs the Linux boot loader which is used to start a Linux system.

mkrescue

mkrescue makes a bootable rescue floppy using the existing kernel and any initial ramdisk.

keytab−lilo.pl

keytab−lilo.pl compiles keytable definitions into a format which lilo can use in order to set the keyboard type during boot.

Lilo Installation Dependencies

Last checked against version 22.1.

Bash: sh Bin86: as86, ld86 Binutils: as, ld, strip Fileutils: cp, dd, ln Gcc: cc, cc1, collect2, cpp0
Make: make Sed: sed Textutils: cat

Linux (the kernel)

Official Download Location

Linux (2.4.19): ftp://ftp.kernel.org/pub/linux/kernel/

Contents of Linux

Last checked against version 2.4.18.

Support Files

The linux kernel and the linux kernel headers

Descriptions

linux kernel

The Linux kernel is at the core of every Linux system. It's what makes Linux tick. When a computer is turned on and boots a Linux system, the very first piece of Linux software that gets loaded is the kernel. The kernel initializes the system's hardware components: serial ports, parallel ports, sound cards, network cards, IDE controllers, SCSI controllers and a lot more. In a nutshell the kernel makes the hardware available so that the software can run.

linux kernel headers

These are the files we copy to `/usr/include/{linux,asm}` in Chapter 5. They should match those which glibc was compiled against and therefore should *not* be replaced when upgrading the kernel. They are essential for compiling many programs.

Linux Installation Dependencies

Last checked against version 2.4.17.

Bash: sh Binutils: ar, as, ld, nm, objcopy Fileutils: cp, ln, mkdir, mv, rm, touch
Findutils: find, xargs Gcc: cc1, collect2, cpp0, gcc Grep: grep Gzip: gzip Make: make Gawk: awk
Modutils: depmod, genksyms Net−tools: dnsdomainname, hostname Sed: sed
Sh−utils: basename, date, expr, pwd, stty, uname, whoami, yes Textutils: cat, md5sum, sort, tail

M4

Official Download Location

M4 (1.4): ftp://ftp.gnu.org/gnu/m4/

Contents of M4

Last checked against version 1.4.

Program Files

m4

Descriptions

m4

m4 is a macro processor. It copies input to output, expanding macros as it goes. Macros are either built−in or user−defined and can take any number of arguments. Besides just doing macro expansion, m4 has built−in functions for including named files, running Unix commands, doing integer arithmetic, manipulating text in various ways, 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.

M4 Installation Dependencies

Last checked against version 1.4.

Bash: sh Binutils: ar, as, ld, ranlib Diffutils: cmp Fileutils: chmod, cp, install, mv, rm Make: make
Gcc: cc1, collect2, cpp0, gcc Grep: egrep, grep Sed: sed Sh−utils: date, echo, hostname
Textutils: cat, tr

Make

Official Download Location

Make (3.79.1): ftp://ftp.gnu.org/gnu/make/

Contents of Make

Last checked against version 3.79.1.

Program files

make

Descriptions

make

make determines automatically which pieces of a large program need to be recompiled, and issues the commands to recompile them.

Make Installation Dependencies

Last checked against version 3.79.1.

Autoconf: autoconf, autoheader Automake: aclocal, automake Bash: sh Binutils: as, ld Diffutils: cmp
Fileutils: chgrp, chmod, install, ls, mv, rm Gcc: cc, cc1, collect2, cpp0, gcc Glibc: getconf
Grep: egrep, fgrep, grep M4: m4 Make: make Gawk: gawk Sed: sed
Sh−utils: basename, echo, expr, hostname, sleep, uname Texinfo: install−info, makeinfo
Textutils: cat, tr

MAKEDEV

Official Download Location

MAKEDEV (1.7): ftp://ftp.linuxfromscratch.org/lfs–packages/4.0–rc1/
http://ftp.linuxfromscratch.org/lfs–packages/4.0–rc1/

Contents of MAKEDEV

Last checked against version 1.5.

Program Files

MAKEDEV

Descriptions

MAKEDEV

`MAKEDEV` is a script that creates the necessary static device nodes usually residing in the `/dev` directory. Detailed information on device nodes can be found in the Linux kernel source tree in `Documentation/devices.txt`.

MAKEDEV Installation Dependencies

Last checked against version 1.5.

Bash: sh Fileutils: chmod, chown, cp, ln, mknod, mv, rm Grep: grep Sh–utils: expr, id

Man

Official Download Location

Man (1.5k): ftp://ftp.win.tue.nl/pub/linux−local/utils/man/ Man Patch (1.5k):
ftp://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/ http://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/

Contents of Man

Last checked against version 1.5k.

Program Files

apropos, makewhatis, man, man2dvi, man2html and whatis

Descriptions

apropos

apropos searches for keywords in a set of database files, containing short descriptions of system commands, and displays the result on the standard output.

makewhatis

makewhatis reads all the manual pages contained in given sections of manpath or the pre−formatted pages contained in the given sections of catpath. For each page, it writes a line in the whatis database. Each line consists of the name of the page and a short description, separated by a dash. The description is extracted using the content of the NAME section of the manual page.

man

man formats and displays the on−line manual pages.

man2dvi

man2dvi converts a manual page into dvi format.

man2html

man2html converts a manual page into html.

whatis

whatis searches for keywords in a set of database files, containing short descriptions of system commands, and displays the result on the standard output. Only complete word matches are displayed.

Man Installation Dependencies

Last checked against version 1.5i2.

Bash: sh Binutils: as, ld Fileutils: chmod, cp, install, mkdir, rm Gcc: c11, collect2, cpp0, gcc Grep: grep Make: make Gawk: awk Sed: sed Sh−utils: echo Textutils: cat

Man−pages

Official Download Location

Man−pages (1.52): ftp://ftp.kernel.org/pub/linux/docs/manpages/

Contents of Man−pages

Last checked against version 1.52.

Support Files

various manual pages that don't come with the packages.

Descriptions

manual pages

Examples of provided manual pages are the manual pages describing all the C and C++ functions, a few important /dev/ files and more.

Man−pages Installation Dependencies

Last checked against version 1.47.

Bash: sh Fileutils: install Make: make

Modutils

Official Download Location

Modutils (2.4.19): ftp://ftp.kernel.org/pub/linux/utils/kernel/modutils/

Contents of Modutils

Last checked against version 2.4.16.

Program Files

depmod, genksyms, insmod, insmod_ksymoops_clean, kallsyms (link to insmod), kernelversion, ksyms (link to insmod), lsmod (link to insmod), modinfo, modprobe (link to insmod) and rmmod (link to insmod)

Descriptions

depmod

depmod handles dependency descriptions for loadable kernel modules.

genksyms

genksyms reads (on standard input) the output from gcc −E source.c and generates a file containing version information.

insmod

insmod installs a loadable module in the running kernel.

insmod_ksymoops_clean

insmod_ksymoops_clean deletes saved ksyms and modules not accessed in 2 days.

kallsyms

kallsyms extracts all kernel symbols for debugging.

kernelversion

kernelversion reports the major version of the running kernel.

ksyms

ksyms displays exported kernel symbols.

lsmod

lsmod shows information about all loaded modules.

modinfo

modinfo examines an object file associated with a kernel module and displays any information that it can glean.

modprobe

modprobe uses a Makefile−like dependency file, created by depmod, to automatically load the relevant module(s) from the set of modules available in predefined directory trees.

rmmod

rmmod unloads loadable modules from the running kernel.

Modutils Installation Dependencies

Last checked against version 2.4.12.

Bash: sh Binutils: ar, as, ld, ranlib, strip Bison: bison Diffutils: cmp
Fileutils: chmod, install, ln, mkdir, mv, rm Flex: flex Gcc: cc, cc1, collect2, cpp0, gcc
Grep: egrep, grep Make: make Sed: sed Sh−utils: basename, expr, hostname, uname Textutils: cat, tr

Ncurses

Official Download Location

Ncurses (5.2): ftp://ftp.gnu.org/gnu/ncurses/ Ncurses Patch (5.2): ftp://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/ http://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/

Contents of Ncurses

Last checked against version 5.2.

Program Files

captoinfo (link to tic), clear, infocmp, infotocap (link to tic), reset (link to tset), tack, tic, toe, tput and tset.

Descriptions

captoinfo

captoinfo converts a termcap description into a terminfo description.

clear

clear clears the screen if this is possible. It looks in the environment for the terminal type and then in the terminfo database to figure out how to clear the screen.

infocmp

infocmp can be used to compare a binary terminfo entry with other terminfo entries, rewrite a terminfo description to take advantage of the use= terminfo field, or print out a terminfo description from the binary file (term) in a variety of formats (the opposite of what tic does).

infotocap

info to cap converts a terminfo description into a termcap description.

reset

reset sets cooked and echo modes, turns off cbreak and raw modes, turns on new−line translation and resets any unset special characters to their default values before doing terminal initialization the same way as tset.

tack

tack is the terminfo action checker.

tic

tic is the terminfo entry−description compiler. The program translates a terminfo file from source format into the binary format for use with the ncurses library routines. Terminfo files contain information about the capabilities of a terminal.

toe

toe lists all available terminal types by primary name with descriptions.

tput

tput uses the terminfo database to make the values of terminal−dependent capabilities and information available to the shell, to initialize or reset the terminal, or return the long name of the requested terminal type.

tset

tset initializes terminals so they can be used, but it's not widely used anymore. It's provided for 4.4BSD compatibility.

Library Files

libcurses.[a,so] (link to libncurses.[a,so]), libform.[a,so], libform_g.a, libmenu.[a,so], libmenu_g.a, libncurses++.a, libncurses.[a,so], libncurses_g.a, libpanel.[a,so] and libpanel_g.a

libcurses, libncurses++, libncurses, libncurses_g

These libraries are the base of the system and are used to display text (often in a fancy way) on the screen. An example where ncurses is used is in the kernel's "make menuconfig" process.

libform, libform_g

libform is used to implement forms in ncurses.

libmenu, libmenu_g

libmenu is used to implement menus in ncurses.

libpanel, libpanel_g

libpanel is used to implement panels in ncurses.

Ncurses Installation Dependencies

Last checked against version 5.2.

Bash: sh Binutils: ar, as, ld, ranlib Diffutils: cmp Fileutils: chmod, cp, install, ln, mkdir, mv, rm
Gcc: c++, cc1, cc1plus, collect2, cpp0, gcc Glibc: ldconfig Grep: egrep, fgrep, grep Make: make
Gawk: gawk Sed: sed Sh−utils: basename, date, echo, expr, hostname, uname
Textutils: cat, sort, tr, wc

Netkit−base

Official Download Location

Netkit−base (0.17): ftp://ftp.uk.linux.org/pub/linux/Networking/netkit/

Contents of Netkit−base

Last checked against version 0.17.

Program Files

inetd and ping

Descriptions

inetd

inetd is the mother of all daemons. It listens for connections, and transfers the call to the appropriate daemon.

ping

ping sends ICMP ECHO_REQUEST packets to a host and determines its response time.

Netkit−base Installation Dependencies

Last checked against version 0.17.

Bash: sh Binutils: as, ld, strip Fileutils: cp, install, rm Make: make Gcc: cc1, collect2, cpp0, gcc
Sed: sed Sh−utils: date Textutils: cat

Net−tools

Official Download Location

Net−tools (1.60): http://www.tazenda.demon.co.uk/phil/net−tools/

Contents of Net−tools

Last checked against version 1.60.

Program Files

arp, dnsdomainname (link to hostname), domainname (link to hostname), hostname, ifconfig, nameif, netstat, nisdomainname (link to hostname), plipconfig, rarp, route, slattach and ypdomainname (link to hostname)

Descriptions

arp

arp is used to manipulate the kernel's ARP cache, usually to add or delete an entry, or to dump the ARP cache.

dnsdomainname

dnsdomainname shows the system's DNS domain name.

domainname

domainname shows or sets the system's NIS/YP domain name.

hostname

hostname prints or sets the name of the current host system.

ifconfig

The ifconfig command is the general command used to configure network interfaces.

nameif

nameif names network interfaces based on MAC addresses.

netstat

netstat is a multi−purpose tool used to print the network connections, routing tables, interface statistics, masquerade connections and multicast memberships.

nisdomainname

nisdomainname shows or sets system's NIS/YP domain name.

plipconfig

plipconfig is used to fine−tune the PLIP device parameters, hopefully making it faster.

rarp

Akin to the arp program, the rarp program manipulates the system's RARP table.

route

route is the general utility which is used to manipulate the IP routing table.

slattach

slattach attaches a network interface to a serial line, i.e.. puts a normal terminal line into one of several "network" modes.

ypdomainname

ypdomainname shows or sets the system's NIS/YP domain name.

Net−tools Installation Dependencies

Last checked against version 1.60.

Bash: bash, sh Binutils: ar, as, ld Fileutils: install, ln, ls, mv, rm Gcc: cc, cc1, collect2, cpp0
Make: make Sh−utils: echo

Patch

Official Download Location

Patch (2.5.4): ftp://ftp.gnu.org/gnu/patch/

Contents of Patch

Last checked against version 2.5.4.

Program Files

patch

Descriptions

patch

The patch program modifies a file according to a patch file. A patch file usually is a list created by the diff program that contains instructions on how an original file needs to be modified. Patch is used a lot for source code patches since it saves time and space. Imagine a package that is 1 MB in size. The next version of that package only has changes in two files of the first version. It can be shipped as an entirely new package of 1 MB or just as a patch file of 1 KB which will update the first version to make it identical to the second version. So if the first version was downloaded already, a patch file avoids a second large download.

Patch Installation Dependencies

Last checked against version 2.5.4.

Bash: sh Binutils: as, ld Diffutils: cmp Fileutils: chmod, install, mv, rm
Gcc: cc, cc1, collect2, cpp0, gcc Glibc: getconf Grep: egrep, grep Make: make Sed: sed
Sh–utils: echo, expr, hostname, uname Textutils: cat, tr

Perl

Official Download Location

Perl (5.8.0): http://www.perl.com/

Contents of Perl

Last checked against version 5.6.1.

Program Files

a2p, c2ph, dprofpp, find2perl, h2ph, h2xs, perl, perl5.6.1, perlbug, perlcc, perldoc, pl2pm, pod2html, pod2latex, pod2man, pod2text, pod2usage, podchecker, podselect, pstruct, s2p and splain

Descriptions

a2p

a2p is an awk to perl translator.

c2ph

c2ph dumps C structures as generated from "cc −g −S" stabs.

dprofpp

dprofpp displays perl profile data.

find2perl

find2perl translates find command lines to Perl code.

h2ph

h2ph converts .h C header files to .ph Perl header files.

h2xs

h2xs converts .h C header files to Perl extensions.

perl, perl5.6.1

perl is the Practical Extraction and Report Language. It combines some of the best features of C, sed, awk and sh into one powerful language.

perlbug

perlbug helps to generate bug reports about perl or the modules that come with it, and mail them.

perlcc

perlcc generates executables from Perl programs.

perldoc

perldoc looks up a piece of documentation in .pod format that is embedded in the perl installation tree or in a perl script and displays it via "pod2man | nroff −man | $PAGER".

pl2pm

pl2pm is a tool to aid in the conversion of Perl4−style .pl library files to Perl5−style library modules.

pod2html

pod2html converts files from pod format to HTML format.

pod2latex

pod2latex converts files from pod format to LaTeX format.

pod2man

pod2man converts pod data to formatted *roff input.

pod2text

pod2text converts pod data to formatted ASCII text.

pod2usage

pod2usage prints usage messages from embedded pod docs in files.

podchecker

podchecker checks the syntax of pod format documentation files.

podselect

podselect prints selected sections of pod documentation on standard output.

pstruct

pstruct dumps C structures as generated from "cc −g −S" stabs.

s2p

s2p is a sed to perl translator.

splain

splain is a program to force verbose warning diagnostics in perl.

Library Files

attrs.so, B.so, ByteLoader.so, DProf.so, Dumper.so, DynaLoader.a, Fcntl.so, Glob.so, Hostname.so, IO.so, libperl.a, Opcode.so, Peek.so, POSIX.so, re.so, SDBM_File.so, Socket.so, Syslog.so and SysV.so

Descriptions

attrs

No description is currently available.

B

No description is currently available.

ByteLoader

No description is currently available.

DProf

No description is currently available.

Dumper

No description is currently available.

DynaLoader

No description is currently available.

Fcntl

No description is currently available.

Glob

No description is currently available.

Hostname

No description is currently available.

IO

No description is currently available.

libperl

No description is currently available.

Opcode

No description is currently available.

Peek

No description is currently available.

POSIX

No description is currently available.

re

No description is currently available.

SDBM_File

No description is currently available.

Socket

No description is currently available.

Syslog

No description is currently available.

SysV

No description is currently available.

Perl Installation Dependencies

Last checked against version 5.6.1.

Bash: sh Binutils: ar, as, ld, nm Diffutils: cmp Fileutils: chmod, cp, ln, ls, mkdir, mv, rm, touch Gcc: cc, cc1, collect2, cpp0, gcc Grep: egrep, grep Make: make Gawk: awk Sed: sed

Sh–utils: basename, date, echo, expr, hostname, pwd, uname, whoami
Textutils: cat, comm, sort, split, tr, uniq, wc

Procinfo

Official Download Location

Procinfo (18): ftp://ftp.cistron.nl/pub/people/svm/

Contents of Procinfo

Last checked against version 18.

Program Files

lsdev, procinfo and socklist

Descriptions

lsdev

lsdev gathers information about your computer's installed hardware from the interrupts, ioports and dma files in the /proc directory, thus giving you a quick overview of which hardware uses what I/O addresses and what IRQ and DMA channels.

procinfo

procinfo gathers some system data from the /proc directory and prints it nicely formatted on the standard output device.

socklist

is a Perl script that gives you a list of all open sockets, enumerating types, port, inode, uid, pid, fd and the program to which it belongs.

Procinfo Installation Dependencies

Last checked against version 18.

Binutils: as, ld Fileutils: install, mkdir Gcc: cc1, collect2, cpp0, gcc Make: make

Procps

Official Download Location

Procps (2.0.7): ftp://people.redhat.com/johnsonm/procps/ Procps Patch (2.0.7): ftp://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/ http://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/

Contents of Procps

Last checked against version 2.0.7.

Program Files

free, kill, oldps, pgrep, pkill, ps, skill, snice, sysctl, tload, top, vmstat, w and watch

Descriptions

free

free displays the total amount of free and used physical and swap memory in the system, as well as the shared memory and buffers used by the kernel.

kill

kills sends signals to processes.

oldps and ps

ps gives a snapshot of the current processes.

pgrep

pgrep looks up processes based on name and other attributes.

pkill

pkill signals processes based on name and other attributes.

skill

skill sends signals to process matching a criteria.

snice

snice changes the scheduling priority for process matching a criteria.

sysctl

sysctl modifies kernel parameters at runtime.

tload

tload prints a graph of the current system load average to the specified tty (or the tty of the tload process if none is specified).

top

top provides an ongoing look at processor activity in real time.

vmstat

vmstat reports information about processes, memory, paging, block IO, traps, and cpu activity.

w

w displays information about the users currently on the machine, and their processes.

watch

watch runs command repeatedly, displaying its output (the first screen full).

Library Files

libproc.so

Descriptions

libproc

libproc is the library against which most of the programs in this set are linked to save disk space by implementing common functions only once.

Procps Installation Dependencies

Last checked against version 2.0.7.

Bash: sh Binutils: as, ld, strip Fileutils: install, ln, mv, rm Gcc: cc1, collect2, cpp0, gcc Grep: grep Make: make Gawk: awk Sed: sed Sh−utils: basename, pwd Textutils: sort, tr

Psmisc

Official Download Location

Psmisc (21): http://download.sourceforge.net/psmisc/
ftp://download.sourceforge.net/pub/sourceforge/psmisc/

Contents of Psmisc

Last checked against version 21.

Program Files

fuser, killall and pstree

Note that in LFS we don't install the pidof link by default because we use pidof from sysvinit instead.

Descriptions

fuser

fuser displays the PIDs of processes that use the specified files or file systems.

killall

killall sends a signal to all processes running any of the specified commands.

pstree

pstree shows running processes as a tree.

Psmisc Installation Dependencies

Last checked against version 20.2.

Autoconf: autoconf, autoheader Automake: aclocal, automake Bash: sh Bison: bison Binutils: as, ld
Diffutils: cmp Fileutils: chmod, install, ls, mkdir, mv, rm Gettext: msgfmt, xgettext
Gcc: cc, cc1, collect2, cpp0, gcc Grep: egrep, grep M4: m4 Make: make Gawk: gawk Sed: sed
Sh–utils: basename, echo, expr, hostname, sleep, uname Texinfo: makeinfo Textutils: cat, tr

Sed

Official Download Location

Sed (3.02): ftp://ftp.gnu.org/gnu/sed/

Contents of Sed

Last checked against version 3.02.

Program Files

sed

Descriptions

sed

sed is a stream editor. A stream editor is used to perform basic text transformations on an input stream (a file or input from a pipeline).

Sed Installation Dependencies

Last checked against version 3.02.

Autoconf: autoconf, autoheader Automake: aclocal, automake Bash: sh Binutils: ar, as, ld, ranlib
Diffutils: cmp Fileutils: chmod, install, ls, mv, rm Gcc: cc1, collect2, cpp0, gcc Glibc: getconf
Grep: egrep, fgrep, grep M4: m4 Make: make Gawk: gawk Sed: sed
Sh−utils: echo, expr, hostname, sleep Texinfo: install−info, makeinfo Textutils: cat, tr

Shadow

Official Download Location

Shadow (4.0.3): ftp://ftp.pld.org.pl/software/shadow/

Contents of Shadow

Last checked against version 4.0.3.

Program Files

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

Descriptions

chage

chage changes the number of days between password changes and the date of the last password change.

chfn

chfn changes a user's full name and other information (office room number, office phone number, and home phone number).

chpasswd

chpasswd reads a file of user name and password pairs from standard input and uses this information to update a group of existing users.

chsh

chsh changes the user login shell.

dpasswd

dpasswd adds, deletes, and updates dial−up passwords for user login shells.

expiry

expiry checks and enforces a password expiration policy.

faillog

faillog formats the contents of the failure log,/var/log/faillog, and maintains failure counts and limits.

gpasswd

gpasswd is used to administer the /etc/group file.

groupadd

The groupadd command creates a new group account using the values specified on the command line and the default values from the system.

groupdel

The groupdel command modifies the system account files, deleting all entries that refer to group.

groupmod

The groupmod command modifies the system account files to reflect the changes that are specified on the command line.

groups

groups prints the groups which a user is in.

grpck

grpck verifies the integrity of the system authentication information.

grpconv

grpunconv converts to shadow group files from normal group files.

grpunconv

grpunconv converts from shadow group files to normal group files.

lastlog

lastlog formats and prints the contents of the last login log, /var/log/lastlog. The login−name, port, and last login time will be printed.

login

login is used to establish a new session with the system.

logoutd

logoutd enforces the login time and port restrictions specified in /etc/porttime.

mkpasswd

mkpasswd reads a file in the format given by the flags and converts it to the corresponding database file format.

newgrp

newgrp is used to change the current group ID during a login session.

newusers

newusers reads a file of user name and clear text password pairs and uses this information to update a group of existing users or to create new users.

passwd

passwd changes passwords for user and group accounts.

pwck

pwck verifies the integrity of the password files.

pwconv

pwconv converts the normal password file to a shadowed password file.

pwunconv

pwunconv converts a shadowed password file to a normal password file.

sg

sg sets the user's GID to that of the given group, or executes a given command as member of the given group.

useradd

useradd creates a new user or update default new user information.

userdel

userdel modifies the system account files, deleting all entries that refer to a specified login name.

usermod

usermod modifies the system account files to reflect the changes that are specified on the command line.

vipw and vigr

vipw and vigr will edit the files /etc/passwd and /etc/group, respectively. With the –s flag, they will edit the shadow versions of those files, /etc/shadow and /etc/gshadow, respectively.

Library Files

libmisc.[a,so], libshadow.[a,so]

Descriptions

libmisc

No description is currently available.

libshadow

libshadow provides common functionality for the shadow programs.

Shadow Installation Dependencies

Last checked against version 20001016.

Autoconf: autoconf, autoheader Automake: aclocal, automake Bash: sh Binutils: ar, as, ld, nm, ranlib Diffutils: cmp Fileutils: chmod, cp, install, ln, ls, mkdir, mv, rm, rmdir Gettext: msgfmt, xgettext Gcc: cc1, collect2, cpp0, gcc Glibc: ldconfig Grep: egrep, grep M4: m4 Make: make Gawk: gawk Net−tools: hostname Sed: sed Sh−utils: basename, echo, expr, sleep, uname Texinfo: makeinfo Textutils: cat, sort, tr, uniq

Sh−utils

Official Download Location

Sh−utils (2.0): ftp://ftp.gnu.org/gnu/sh−utils/ Sh−utils Patch (2.0): ftp://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/ http://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/

Contents of Sh−utils

Last checked against version 2.0.

Program Files

basename, chroot, date, dirname, echo, env, expr, factor, false, groups, hostid, id, logname, nice, nohup, pathchk, pinky, printenv, printf, pwd, seq, sleep, stty, su, tee, test, true, tty, uname, uptime, users, who, whoami and yes

Descriptions

basename

basename strips directory and suffixes from filenames.

chroot

chroot runs a command or interactive shell with special root directory.

date

date displays the current time in a specified format, or sets the system date.

dirname

dirname strips non−directory suffixes from file name.

echo

echo displays a line of text.

env

env runs a program in a modified environment.

expr

expr evaluates expressions.

factor

factor prints the prime factors of all specified integer numbers.

false

false always exits with a status code indicating failure.

groups

groups prints a user's group memberships.

hostid

hostid prints the numeric identifier (in hexadecimal) for the current host.

id

id prints the effective user and group IDs of the current user or a given user.

logname

logname prints the current user's login name.

nice

nice runs a program with modified scheduling priority.

nohup

nohup runs a command immune to hangups, with output to a log file.

pathchk

pathchk checks whether file names are valid or portable.

pinky

pinky is a lightweight finger utility which retrieves information about a certain user.

printenv

printenv prints all or part of the environment.

printf

printf formats and prints data (the same as the C printf function).

pwd

pwd prints the name of the current/working directory.

seq

seq prints numbers in a certain range with a certain increment.

sleep

sleep delays for a specified amount of time.

stty

stty changes and prints terminal line settings.

su

su runs a shell with substitute user and group IDs.

tee

tee reads from standard input and writes to standard output and files.

test

test checks file types and compares values.

true

true always exits with a status code indicating success.

tty

tty prints the file name of the terminal connected to standard input.

uname

uname prints system information.

uptime

uptime tells how long the system has been running.

users

users prints the user names of users currently logged in to the current host.

who

who shows who is logged on.

whoami

whoami prints the user name associated with the current effective user ID.

yes

yes outputs 'y' or a given string repeatedly, until killed.

Sh−utils Installation Dependencies

Last checked against version 2.0.

Autoconf: autoconf, autoheader Automake: aclocal, automake Bash: sh Binutils: ar, as, ld, ranlib Diffutils: cmp Fileutils: chmod, chown, install, ls, mv, rm Gettext: msgfmt, xgettext Gcc: cc, cc1, collect2, cpp0, gcc Glibc: getconf Grep: egrep, fgrep, grep M4: m4 Make: make Gawk: gawk Perl: perl Sed: sed Sh−utils: basename, echo, expr, hostname, sleep, uname Tar: tar Texinfo: install−info, makeinfo Textutils: cat, tr

Sysklogd

Official Download Location

Sysklogd (1.4.1): http://www.infodrom.org/projects/sysklogd/

Contents of Sysklogd

Last checked against version 1.4.1.

Program Files

klogd and syslogd

Descriptions

klogd

klogd is a system daemon which intercepts and logs Linux kernel messages.

syslogd

syslogd provides the kind of logging that many modern programs use. Every logged message contains at least a time and a hostname field, normally a program name field, too, but that depends on how trusty the logging program is.

Sysklogd Installation Dependencies

Last checked against version 1.4.1.

Binutils: as, ld, strip Fileutils: install Gcc: cc1, collect2, cpp0, gcc Make: make

Sysvinit

Official Download Location

Sysvinit (2.84): ftp://ftp.cistron.nl/pub/people/miquels/sysvinit/

Contents of Sysvinit

Last checked against version 2.84.

Program Files

halt, init, killall5, last, lastb (link to last), mesg, pidof (link to killall5), poweroff (link to halt), reboot (link to halt), runlevel, shutdown, sulogin, telinit (link to init), utmpdump and wall

Descriptions

halt

halt notes that the system is being brought down in the file /var/log/wtmp, and then either tells the kernel to halt, reboot or poweroff the system. If halt or reboot is called when the system is not in runlevel 0 or 6, shutdown will be invoked instead (with the flag −h or −r).

init

init is the parent of all processes. Its primary role is to create processes from a script stored in the file /etc/inittab. This file usually has entries which cause init to spawn gettys on each line that users can log in. It also controls autonomous processes required by any particular system.

killall5

killall5 is the SystemV killall command. It sends a signal to all processes except the processes in its own session, so it won't kill the shell that is running the script it was called from.

last

last searches back through the file /var/log/wtmp (or the file designated by the −f flag) and displays a list of all users logged in (and out) since that file was created.

lastb

lastb is the same as last, except that by default it shows a log of the file /var/log/btmp, which contains all the bad login attempts.

mesg

mesg controls the access to the user's terminal by others. It's typically used to allow or disallow other users to write to his terminal.

pidof

pidof displays the process identifiers (PIDs) of the named programs.

poweroff

poweroff is equivalent to shutdown −h −p now. It halts the computer and switches off the computer (when using an APM compliant BIOS and APM is enabled in the kernel).

reboot

reboot is equivalent to shutdown −r now. It reboots the computer.

runlevel

runlevel reads the system utmp file (typically /var/run/utmp) to locate the runlevel record, and then prints the previous and current system runlevel on its standard output, separated by a single space.

shutdown

shutdown brings the system down in a secure way. All logged−in users are notified that the system is going down, and login is blocked.

sulogin

sulogin is invoked by init when the system goes into single user mode (this is done through an entry in /etc/inittab). Init also tries to execute sulogin when it is passed the −b flag from the boot loader (LILO, for example).

telinit

telinit sends appropriate signals to init, telling it which runlevel to change to.

utmpdump

utmpdumps prints the content of a file (usually /var/run/utmp) on standard output in a user friendly format.

wall

wall sends a message to everybody logged in with their mesg permission set to yes.

Sysvinit Installation Dependencies

Last checked against version 2.84.

Bash: sh Binutils: as, ld Fileutils: chown, cp, install, ln, mknod, rm Gcc: cc, cc1, collect2, cpp0
Make: make Sed: sed

Tar

Official Download Location

Tar (1.13): ftp://ftp.gnu.org/gnu/tar/ Tar Patch (1.13): ftp://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/ http://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/

Contents of Tar

Last checked against version 1.13.

Program Files

rmt and tar

Descriptions

rmt

rmt is a program used by the remote dump and restore programs in manipulating a magnetic tape drive through an interprocess communication connection.

tar

tar is an archiving program designed to store and extract files from an archive file known as a tar file.

Tar Installation Dependencies

Last checked against version 1.13.

Autoconf: autoconf, autoheader Automake: aclocal, automake Bash: sh Binutils: ar, as, ld, ranlib
Diffutils: cmp Fileutils: chmod, install, ls, mv, rm Gettext: msgfmt, xgettext
Gcc: cc, cc1, collect2, cpp0, gcc Glibc: getconf Grep: egrep, fgrep, grep M4: m4 Make: make
Gawk: gawk Net−tools: hostname Patch: patch Sed: sed Sh−utils: basename, echo, expr, sleep, uname
Texinfo: install−info, makeinfo Textutils: cat, tr

Texinfo

Official Download Location

Texinfo (4.2): ftp://ftp.gnu.org/gnu/texinfo/

Contents of Texinfo

Last checked against version 4.2.

Program Files

info, infokey, install−info, makeinfo, texi2dvi and texindex

Descriptions

info

The info program reads Info documents, usually contained in the /usr/share/info directory. Info documents are like man(ual) pages, but they tend to go deeper than just explaining the options to a program.

infokey

infokey compiles a source file containing Info customizations into a binary format.

install−info

The install−info program updates the info entries. When the info program is run, a list with available topics (ie: available info documents) will be presented. The install−info program is used to maintain this list of available topics. If info files are removed manually, you must also delete the topic in the index file. This program is used for that. It also works the other way around when info documents are added.

makeinfo

The makeinfo program translates Texinfo source documents into various formats. Available formats are: info files, plain text and HTML.

texi2dvi

The texi2dvi program prints Texinfo documents.

texindex

The texindex program is used to sort Texinfo index files.

Texinfo Installation Dependencies

Last checked against version 4.0.

Bash: sh Binutils: ar, as, ld, ranlib Diffutils: cmp Fileutils: chmod, install, ln, ls, mkdir, mv, rm
Gcc: cc1, collect2, cpp0, gcc Grep: egrep, fgrep, grep Make: make Sed: sed
Sh−utils: basename, echo, expr, hostname, sleep Texinfo: makeinfo Textutils: cat, tr

Textutils

Official Download Location

Textutils (2.1): ftp://ftp.gnu.org/gnu/textutils/

Contents of Textutils

Last checked against version 2.0.

Program Files

cat, cksum, comm, csplit, cut, expand, fmt, fold, head, join, md5sum, nl, od, paste, pr, ptx, sort, split, sum, tac, tail, tr, tsort, unexpand, uniq and wc

Descriptions

cat

cat concatenates file(s) or standard input to standard output.

cksum

cksum prints CRC checksum and byte counts of each specified file.

comm

comm compares two sorted files line by line.

csplit

csplit outputs pieces of a file separated by (a) pattern(s) to files xx01, xx02, ..., and outputs byte counts of each piece to standard output.

cut

cut prints selected parts of lines from specified files to standard output.

expand

expand converts tabs in files to spaces, writing to standard output.

fmt

fmt reformats each paragraph in the specified file(s), writing to standard output.

fold

fold wraps input lines in each specified file (standard input by default), writing to standard output.

head

head prints the first xx (10 by default) lines of each specified file to standard output.

join

join joins lines of two files on a common field.

md5sum

md5sum prints or checks MD5 checksums.

nl

nl writes each specified file to standard output, with line numbers added.

od

od writes an unambiguous representation, octal bytes by default, of a specified file to standard output.

paste

paste writes lines consisting of the sequentially corresponding lines from each specified file, separated by TABs, to standard output.

pr

pr paginates or columnates files for printing.

ptx

ptx produces a permuted index of file contents.

sort

sort writes sorted concatenation of files to standard output.

split

split outputs fixed−size pieces of an input file to PREFIXaa, PREFIXab, ...

sum

sum prints checksum and block counts for each specified file.

tac

tac writes each specified file to standard output, last line first.

tail

tail print the last xx (10 by default) lines of each specified file to standard output.

tr

tr translates, squeezes, and/or deletes characters from standard input, writing to standard output.

tsort

tsort writes totally ordered lists consistent with the partial ordering in specified files.

unexpand

unexpand converts spaces in each file to tabs, writing to standard output.

uniq

uniq removes duplicate lines from a sorted file.

wc

wc prints line, word and byte counts for each specified file and a total line, if more than one file is specified.

Textutils Installation Dependencies

Last checked against version 2.0.

Autoconf: autoconf, autoheader Automake: aclocal, automake Bash: sh Binutils: ar, as, ld, ranlib
Diffutils: cmp Fileutils: chmod, install, ls, mv, rm Gettext: msgfmt, xgettext
Gcc: cc, cc1, collect2, cpp0, gcc Glibc: getconf Grep: egrep, fgrep, grep M4: m4 Make: make
Gawk: gawk Net−tools: hostname Perl: perl Sed: sed Sh−utils: basename, echo, expr, sleep, uname
Tar: tar Texinfo: install−info, makeinfo Textutils: cat, tr

Util−linux

Official Download Location

Util−linux (2.11u): ftp://ftp.win.tue.nl/pub/linux−local/utils/util−linux/

Contents of Util−linux

Last checked against version 2.11t.

Program Files

agetty, arch, blockdev, cal, cfdisk, chkdupexe, col, colcrt, colrm, column, ctrlaltdel, cytune, ddate, dmesg, elvtune, fdformat, fdisk, fsck.cramfs, fsck.minix, getopt, hexdump, hwclock, ipcrm, ipcs, isosize, line, logger, look, losetup, mcookie, mkfs, mkfs.bfs, mkfs.cramfs, mkfs.minix, mkswap, more, mount, namei, parse.bash, parse.tcsh, pg, pivot_root, ramsize (link to rdev), raw, rdev, readprofile, rename, renice, rev, rootflags (link to rdev), script, setfdprm, setsid, setterm, sfdisk, swapoff (link to swapon), swapon, test.bash, test.tcsh, tunelp, ul, umount, vidmode (link to rdev), whereis and write

Descriptions

agetty

agetty opens a tty port, prompts for a login name and invokes the /bin/login command.

arch

arch prints the machine architecture.

blockdev

blockdev allows to call block device ioctls from the command line.

cal

cal displays a simple calender.

cfdisk

cfdisk is a libncurses based disk partition table manipulator.

chkdupexe

chkdupexe finds duplicate executables.

col

col filters reverse line feeds from input.

colcrt

colcrt filters nroff output for CRT previewing.

colrm

colrm removes columns from a file.

column

column columnates lists.

ctrlaltdel

ctrlaltdel sets the function of the CTRL+ALT+DEL key combination (hard or soft reset).

cytune

cytune queries and modifies the interruption threshold for the Cyclades driver.

ddate

ddate converts Gregorian dates to Discordian dates.

dmesg

dmesg is used to examine or control the kernel ring buffer (boot messages from the kernel).

elvtune

elvtune allows to tune the I/O elevator per block device queue basis.

fdformat

fdformat low−level formats a floppy disk.

fdisk

fdisk is a disk partition table manipulator.

fsck.cramfs

No description is currently available.

fsck.minix

fsck.minix performs a consistency check for the Linux MINIX filesystem.

getopt

getops parses command options the same way as the getopt C command.

hexdump

hexdump displays specified files, or standard input, in a user specified format (ascii, decimal, hexadecimal, octal).

hwclock

hwclock queries and sets the hardware clock (Also called the RTC or BIOS clock).

ipcrm

ipcrm removes a specified resource.

ipcs

ipcs provides information on IPC facilities.

isosize

isosize outputs the length of a iso9660 file system.

line

line copies one line (up to a newline) from standard input and writes it to standard output.

logger

logger makes entries in the system log.

look

look displays lines beginning with a given string.

losetup

losetup sets up and controls loop devices.

mcookie

mcookie generates magic cookies for xauth.

mkfs

mkfs builds a Linux filesystem on a device, usually a harddisk partition.

mkfs.bfs

mkfs.bfs creates a SCO bfs file system on a device, usually a harddisk partition.

mkfs.cramfs

No description is currently available.

mkfs.minix

mkfs.minix creates a Linux MINIX filesystem on a device, usually a harddisk partition.

mkswap

mkswap sets up a Linux swap area on a device or in a file.

more

more is a filter for paging through text one screen full at a time.

mount

mount mounts a filesystem from a device to a directory (mount point).

namei

namei follows a pathname until a terminal point is found.

parse.bash, parse.tcsh, test.bash, test.tcsh

These are example scripts for using the getopt program with either BASH or TCSH.

pg

No description is currently available.

pivot_root

pivot_root moves the root file system of the current process.

ramsize

ramsize queries and sets RAM disk size.

raw

raw is used to bind a Linux raw character device to a block device.

rdev

rdev queries and sets image root device, swap device, RAM disk size, or video mode.

readprofile

readprofile reads kernel profiling information.

rename

rename renames files.

renice

renice alters priority of running processes.

rev

rev reverses lines of a file.

rootflags

rootflags queries and sets extra information used when mounting root.

script

script makes typescript of terminal session.

setfdprm

setfdprm sets user–provides floppy disk parameters.

setsid

setsid runs programs in a new session.

setterm

setterm sets terminal attributes.

sfdisk

sfdisk is a disk partition table manipulator.

swapoff

swapoff disables devices and files for paging and swapping.

swapon

swapon enables devices and files for paging and swapping.

tunelp

tunelp sets various parameters for the LP device.

ul

ul reads a file and translates occurrences of underscores to the sequence which indicates underlining for the terminal in use.

umount

umount unmounts a mounted filesystem.

vidmode

vidmode queries and sets the video mode.

whereis

whereis locates a binary, source and manual page for a command.

write

write sends a message to another user.

Util−linux Installation Dependencies

Last checked against version 2.11n.

Bash: sh Binutils: as, ld Diffutils: cmp Fileutils: chgrp, chmod, cp, install, ln, mv, rm
Gettext: msgfmt, xgettext Gcc: cc, cc1, collect2, cpp, cpp0 Glibc: rpcgen Grep: grep Make: make
Sed: sed Sh−utils: uname, whoami Textutils: cat

Vim

Official Download Location

Vim (6.1): ftp://ftp.vim.org/pub/editors/vim/unix/ Vim Patch (6.1): ftp://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/ http://ftp.linuxfromscratch.org/lfs−packages/4.0−rc1/

Contents of Vim

Last checked against version 6.1.

Program Files

efm_filter.pl, efm_perl.pl, ex (link to vim), less.sh, mve.awk, pltags.pl, ref, rview (link to vim), rvim (link to vim), shtags.pl, tcltags, vi (link to vim), view (link to vim), vim, vim132, vim2html.pl, vimdiff (link to vim), vimm, vimspell.sh, vimtutor and xxd

Descriptions

efm_filter.pl

efm_filter.pl is a filter which reads from stdin, copies to stdout and creates an error file that can be read by vim.

efm_perl.pl

efm_perl.pl reformats the error messages of the Perl interpreter for use with the quickfix mode of vim.

ex

ex starts vim in Ex mode.

less.sh

less.sh is a script which starts vim with less.vim.

mve.awk

mve.awk processes vim errors.

pltags.pl

pltags.pl creates a tags file for Perl code, for use by vim.

ref

ref checks the spelling of arguments.

rview

rview is a restricted version of view. No shell commands can be started and vim can't be suspended.

rvim

rvim is the restricted version of vim. No shell commands can be started and vim can't be suspended.

shtags.pl

shtags.pl generates a tag file for perl scripts.

tcltags

tcltags generates a tag file for TCL code.

vi

vi starts vim in vi−compatible mode.

view

view starts vim in read−only mode.

vim

vim starts vim in the normal, default way.

vim132

vim132 starts vim with the terminal in 132 column mode.

vim2html.pl

vim2html.pl converts vim documentation to HTML.

vimdiff

vimdiff edits two or three versions of a file with vim and show differences.

vimm

vimm enables the DEC locator input model on a remote terminal.

vimspell.sh

vimspell.sh is a script which spells a file and generates the syntax statements necessary to highlight in vim.

vimtutor

vimtutor starts the Vim tutor.

xxd

xxd makes a hexdump or does the reverse.

Vim Installation Dependencies

Last checked against version 6.0.

Bash: sh Binutils: as, ld, strip Diffutils: cmp, diff Fileutils: chmod, cp, ln, mkdir, mv, rm, touch Find: find Gcc: cc1, collect2, cpp0, gcc Grep: egrep, grep Make: make Net−tools: hostname Sed: sed Sh−utils: echo, expr, uname, whoami Textutils: cat, tr, wc

Zlib

Official Download Location

Zlib (1.1.4): ftp://ftp.info–zip.org/pub/infozip/zlib/

Contents of Zlib

Last checked against version 1.1.4.

Library Files

libz[a,so]

Descriptions

libz

This is the zlib library, which is used by many programs for its compression and uncompression functions.

Zlib Installation Dependencies

No dependencies checked yet.

Appendix B. Resources

Introduction

A list of books, HOWTOs and other documents that might be useful to download or buy follows. This list is just a small list to start with. We hope to be able to expand this list in time as we come across more useful documents or books.

Books

- Linux Network Administrator's Guide published by O'Reilly. ISBN: 1−56502−087−2
- Running Linux published by O'Reilly. ISBN: 1−56592−151−8

HOWTOs and Guides

All of the following HOWTOs can be downloaded from the Linux Documentation Project site at http://www.tldp.org

- Linux Network Administrator's Guide
- From−PowerUp−To−Bash−Prompt−HOWTO

Other

- The various man and info pages that come with the packages